

## A Comparative Tutorial of Bayesian Sequential Design and Reinforcement Learning

Mauricio Tec, Yunshan Duan & Peter Müller

To cite this article: Mauricio Tec, Yunshan Duan & Peter Müller (2023) A Comparative Tutorial of Bayesian Sequential Design and Reinforcement Learning, The American Statistician, 77:2, 223-233, DOI: [10.1080/00031305.2022.2129787](https://doi.org/10.1080/00031305.2022.2129787)

To link to this article: <https://doi.org/10.1080/00031305.2022.2129787>



Published online: 31 Oct 2022.



Submit your article to this journal [↗](#)



Article views: 187



View related articles [↗](#)



View Crossmark data [↗](#)

TUTORIAL



# A Comparative Tutorial of Bayesian Sequential Design and Reinforcement Learning

Mauricio Tec<sup>a</sup> , Yunshan Duan<sup>b</sup>, and Peter Müller<sup>b</sup>

<sup>a</sup>Department of Biostatistics, Harvard T.H. Chan School of Public Health, Cambridge, MA; <sup>b</sup>Department of Statistics and Data Science, The University of Texas at Austin, Austin, TX

## ABSTRACT

Reinforcement learning (RL) is a computational approach to reward-driven learning in sequential decision problems. It implements the discovery of optimal actions by learning from an agent interacting with an environment rather than from supervised data. We contrast and compare RL with traditional sequential design, focusing on simulation-based Bayesian sequential design (BSD). Recently, there has been an increasing interest in RL techniques for healthcare applications. We introduce two related applications as motivating examples. In both applications, the sequential nature of the decisions is restricted to sequential stopping. Rather than a comprehensive survey, the focus of the discussion is on solutions using standard tools for these two relatively simple sequential stopping problems. Both problems are inspired by adaptive clinical trial design. We use examples to explain the terminology and mathematical background that underlie each framework and map one to the other. The implementations and results illustrate the many similarities between RL and BSD. The results motivate the discussion of the potential strengths and limitations of each approach.

## ARTICLE HISTORY

Received May 2022  
Accepted September 2022

## KEY WORDS

Bayesian methods;  
Experimental design;  
Reinforcement learning

## 1. Introduction

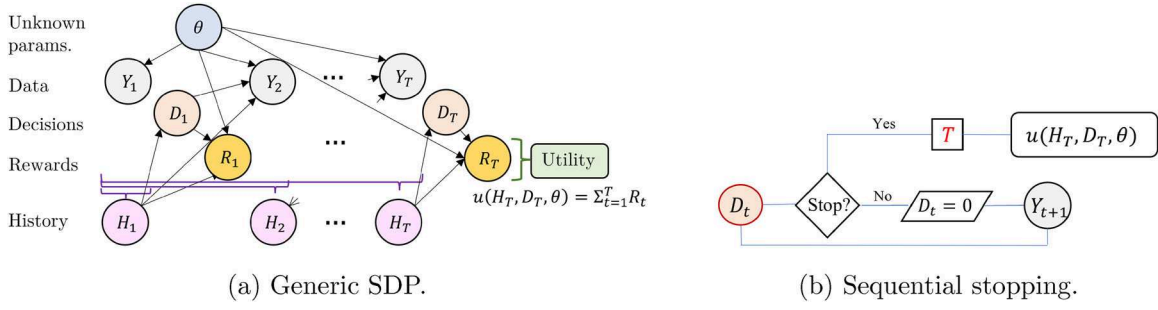
Sequential design problems (SDP) involve a sequence of decisions  $D_t$  with data  $Y_t$  observed at every time step  $t = 1, \dots, T$  (DeGroot 2004; Berger 2013). The goal is to find a decision rule  $(Y_1, \dots, Y_t) \mapsto D_t$  that maximizes the expected value of a utility function. The utility function encodes an agent's preferences as a function of hypothetical future data and assumed truth. The decision rule is prescribed before observing future data beyond  $Y_t$  – as the term “design” emphasizes—assuming a probabilistic model with unknown parameters  $\theta$  that generate the future data. For example,  $\theta$  can be the true effect of a drug. Figure 1(a) summarizes the setup of a general SDP. As motivating examples in the upcoming discussion we will use two examples of clinical trial design, which naturally give rise to SDPs (Berry and Ho 1988; Christen and Nakamura 2003; Rossell, Müller, and Rosner 2007). Both examples are about sequential stopping, that is, the sequential decision is to determine *when* and *how* to end the study. Figure 1(b) shows the setup of sequential stopping problems.

Using these examples, we compare two families of simulation-based methods for solving SDPs with applications in sequential stopping. The first is simulation-based Bayesian Sequential Design (BSD) (Müller et al. 2007), which is based on Bayesian decision theory (Berger 2013). The other approach is Reinforcement Learning (RL), a paradigm based on the interaction between an agent and an environment that results in potential rewards (or costs) for each decision. RL has recently been proposed as a method to implement SDPs focusing on recent advances in deep learning (Shen and Huan 2021), outside the

context of clinical studies. Earlier application of RL related to clinical study design can be found in the dynamic treatment regimes literature (Murphy 2003; Murphy et al. 2007; Zhao, Kosorok, and Zeng 2009). There is a longstanding literature on such problems in statistics; see the review by Parmigiani and Inoue (2009, chap. 15) and the references therein. Implementing RL and BSD in these two motivating example problems (using standard algorithms) will illustrate many similarities between the two frameworks, while highlighting the potential strengths and limitations of each paradigm.<sup>1</sup>

A note about the scope of the upcoming discussion. It is meant to highlight the similarities and differences of algorithms in BSD and RL, and we provide a (partial) mapping of notations. There is no intent to provide (yet another) review of BSD or RL and its many variations. This article focuses on a few variations that best contrast the two traditions, and we try to highlight the relative advantages of each method. In short, BSD is better equipped to deal with additional structure, that is, to include more details in the inference model. For example, dealing with delayed responses in a clinical study, one might want to include a model to use early responses to predict such delayed outcomes. Or one might want to borrow strength across multiple related problems. Doing so would also be possible in RL, but it requires to leave the framework of Markov decision processes underlying many RL algorithms (Gaon and Brafman 2020), and discussed below. On the other hand, RL allows

<sup>1</sup>The implementation code is freely available at <https://github.com/mauriciogtec/bsd-and-rl>.



**Figure 1.** Panel (a) shows the directed graph of a general SDP problem. In the figure,  $H_t = \{Y_1, D_1, Y_2, \dots, D_{t-1}, Y_t\}$  denotes the history or information set at decision time  $t$ . There is an implicit arrow from every element of  $H_t$  onto  $D_t$ , summarily implied by the arrow  $H_t \rightarrow D_t$ . In RL the utility function is usually written as a sum  $G = \sum_{t=1}^T R_t$  of immediate rewards  $R_t(H_t, D_t)$  (although it is usually called the *return* instead of the utility). Panel (b) shows a flow chart for the special case of sequential stopping problems, with  $D_t = 0$  indicating continuation. The node  $T$  represents stopping the trial ( $t = T$  and  $D_T \neq 0$ ).

computation-efficient implementations, and is routinely used for much larger scale problems than BSD. The availability of computation-efficient implementations is critical in applications where sequential designs need to be evaluated for summaries under (hypothetical) repeated experimentation. This is the case, for example, in clinical trial design when frequentist error rates and power need to be reported. The evaluation requires Monte Carlo experiments with massive repeat simulations, making computation-efficient implementation important.

**BSD.** BSD is a model-based paradigm for SDPs based on a sampling model  $p(Y_t \mid H_{t-1}, D_{t-1}, \theta)$  for the observed data and a prior  $p(\theta)$  reflecting the agent’s uncertainty about the unknown parameters.

To compare alternative decisions  $D_t$ , the agent uses an optimality criterion that is formalized as a utility function  $u(Y_1, \dots, Y_T, D_1, \dots, D_T, \theta)$  which quantifies the agent’s preferences under hypothetical data, decisions and truth. It will be convenient to write the utility as  $u(H_T, D_T = d, \theta)$ , where  $H_t$  denotes the history  $H_t := (Y_1, D_1, \dots, D_{t-1}, Y_t)$  at decision time  $t$ . Rational decision makers should act as if they were to maximize such utility in expectation conditioning on already observed data, and marginalizing with respect to any (still) unknown quantities like future data or parameters (DeGroot 2004).

To develop a solution strategy, we start at time  $T$  (final horizon or stopping time). Denote by  $U(H_T, d) = E\{u(H_T, D_T = d, \theta) \mid H_T\}$  the expected utility at the stopping time  $T$ . Then, the rational agent would select  $D_T^*(H_T) = \arg \max_d U(H_T, d)$  at the stopping time. For earlier time steps, rational decisions derive from expectations over future data, with later optimal decisions plugged in,

$$U(H_t, d) = E\{u(H_t, D_t = d, Y_{t+1}(D_t = d), D_{t+1}^*(H_{t+1}(D_t = d)), \dots, D_T^*(H_T(D_t = d)), \theta) \mid H_t\} \quad (1)$$

which determine the optimal decision (“Bayes rule”) as

$$D_t^*(H_t) = \arg \max_d U(H_t, d). \quad (2)$$

Here, we use a *potential outcomes* notation  $H_{t+1}(D_t = d)$  to emphasize that future history depends on the action  $D_t = d$  (Robins 1997), and similarly for other quantities. In the notation for expected utility  $U(\cdot)$ , the lack of an argument indicates marginalization (with respect to future data and parameters)

and optimization (with respect to future actions), respectively. For example, in (1) expected utility conditions on  $H_t$ , but marginalizes w.r.t. future data  $Y_{t+k}$  and substitutes optimal future decisions  $D_{t+k}^*$ .

Some readers, especially those familiar with RL, might wonder why  $D_t = d$  does not appear on the right-hand side of the conditional expectation (as in RL’s state-action value functions). This is because in the BSD framework actions are deterministic. There is no good reason why a rational decision maker would randomize (Berger 2013). However, note that in clinical studies randomization is usually included and desired, but for other reasons—not to achieve an optimal decision, but to facilitate attribution of differences in outcomes to the treatment selection.

**RL.** RL addresses a wider variety of sequential problems than BSD, provided one can formulate them as an agent interacting with an environment yielding rewards  $R_t$  at every time step. Environments can be based on simulations. For example, popular successful RL applications with simulation-based environments include Atari video-games (Mnih et al. 2013), complex board games like chess and Go (Silver et al. 2018), robotic tasks (Tunyasuvunakool et al. 2020) and autonomous driving (Sallab et al. 2017; Wurman et al. 2022).

Just as in BSD, the interactive setup for RL (transition and rewards) can be defined by a sampling model and a prior over  $\theta$ . The interaction replicates the decision process, shown in Figure 1. Each draw  $\theta \sim p(\theta)$  constitutes a new instance or *episode* of the environment. The RL agent seeks to maximize the expected sum of rewards  $G = \sum_{t=1}^T R_t$ , known as the *return*, over an episode. The return  $G$  is the analogue of the utility function.

Decision rules are called *policies* in RL. A (stochastic) policy maps an observed history to a distribution over actions  $D_t \sim \pi(\cdot \mid H_t)$ . The optimal policy  $\pi_*$  satisfies  $\pi_* = \arg \max_{\pi} E\{G \mid \pi\}$ . As mentioned before, the notion of stochastic policies is not natural in BSD with its focus on decisions that a rational agent would take. In fact, under some regularity conditions it can be shown that also the optimal RL policy is deterministic (Puterman 2014). So why stochastic policies? Stochastic policies in RL serve to implement exploration. In BSD it is assumed that if exploration were called for, it would be recognized by the optimal decision rule. While in theory this is the case, in practice

**Table 1.** A brief comparison of key quantities in BSD and RL.

	BSD	RL
$Y_t$		data observed at time $t$
$H_t = (Y_1, D_1 \dots D_{t-1}, Y_t)$		history (information set) at decision time $t$
$D_t = D_t(H_t)$		action (decision) at time $t$
$S_t = S_t(H_t)$	summary (sufficient) statistic	state
$D_{\phi,t}$	(deterministic) action at time $t$	n/a <sup>(a)</sup>
$\pi(D_t   H_t)$	indexed by parameter $\phi$ (policy)	(random) policy
$\pi_\phi$	n/a (no randomization)	policy indexed by $\phi$
optimal decision/policy	Bayes rule $D^*(H_t)$ (2)	optimal policy $\pi_*(H_t)$
$\theta$		unknown parameter
	(usually) required	optional
$R_t$	n/a <sup>(b)</sup>	(immediate) reward at time $t$
optimality criterion	utility $u(Y, D, \theta) = u(H_t, D_t, \theta)$	total return $G = \sum_{k=1}^T R_k$ or remaining return $G_t = \sum_{k=t}^T R_k$
state-action value	n/a (deterministic $D^*$ )	$Q^\pi(H_t, D_t) = E\{G_t   H_t, D_t, \pi\}$
state value	n/a (deterministic $D^*$ )	$V^\pi(H_t) = E\{G_t   H_t, \pi\}$
value under optimal future actions	$U(H_t, D_t)$	$Q^{\pi*}(H_t, D_t)$
optimal value	$U(H_t, D^*(H_t))$	$V^{\pi*}(H_t)$
$J(\phi)$		expectation under policy/decision indexed by $\phi$
	$= E\{u(Y, D_\phi, \theta)\}$	$= E\{G   \pi_\phi\}$
Bellman equation/ backward induction	$U(H_t, D_t) =$ $E\{U(H_{t+1}(D_t))   H_t\}$	$Q^\pi(H_t, D_t) =$ $E\{R_t + V^\pi(H_{t+1})   H_t, D_t, \pi\}$

NOTE: Variations without time subindex  $t$  refer to time-invariant versions. Using states  $S_t$ , in many instances an argument  $H_t$  can be replaced by  $S_t$ , as in  $D_t(S_t)$ . We use  $Y = (Y_1, \dots, Y_T)$  etc. to refer to lists over  $t = 1, \dots, T$ .

<sup>(a)</sup> deterministic policies  $\pi_\phi(H_t)$  are not discussed in this review, but see Silver et al. (2014).

<sup>(b)</sup> additive decomposition as  $u(H_t, D_t, \theta) = \sum_t R(H_t, D_t, \theta)$  is possible, but not usually made explicit.

additional reinforcement of exploration is reasonable. Also, as we shall see later, the use of stochastic policies facilitates the search for optimal policies by allowing the use of the stochastic gradient theorem.

Another close similarity of BSD and RL occurs in the definition of expected utility and the state-action value function in RL. The *state-action value function* is  $Q^\pi(H_t, d) = E\{\sum_{k=t}^T R_k | H_t, D_t = d, \pi\}$ . The value function of the optimal policy  $Q^{\pi*}(H_t, d)$  plays the same role as expected utility  $U(H_t, d)$  when optimal decisions are substituted for future decisions  $D_s$ ,  $s > t$ . An important difference is the stochastic nature of  $\pi$  in the state-value function, versus the deterministic decisions  $D_t$  in BSD.

From this brief introduction one can already notice many correspondences between the objects in RL and BSD. Table 1 shows a partial mapping between their respective terminologies. Not all are perfect equivalences. Sometimes common use in BSD and RL involves different levels of marginalization and/or substituting optimal values. Some of the analogies in the table will be developed in the remainder of the article.

## 2. Two Examples of Optimal Stopping in Clinical Trials

The two stylized examples introduced here mimic sequential stopping in a clinical trial. The agent is an investigator who is planning and overseeing the trial. The data  $Y_t$  are clinical outcomes recorded for each patient. In both cases  $D_t$  refers to a stopping decision after  $t$  (cohorts of) patients. Under continuation ( $D_t = 0$ ), the agent incurs an additional cost  $c_t$  for recruiting the next cohort of patients. Under stopping ( $D_t \neq 0$ ), on the other hand, the agent incurs a cost if a wrong (precise meaning to be specified) recommendation is made. At each time the agent has to choose between continuing the study—to learn

more—versus stopping and realizing a reward for a good final recommendation (about the treatment). Throughout we use the notions of cost (or loss) and utility interchangeably, treating loss as negative utility.

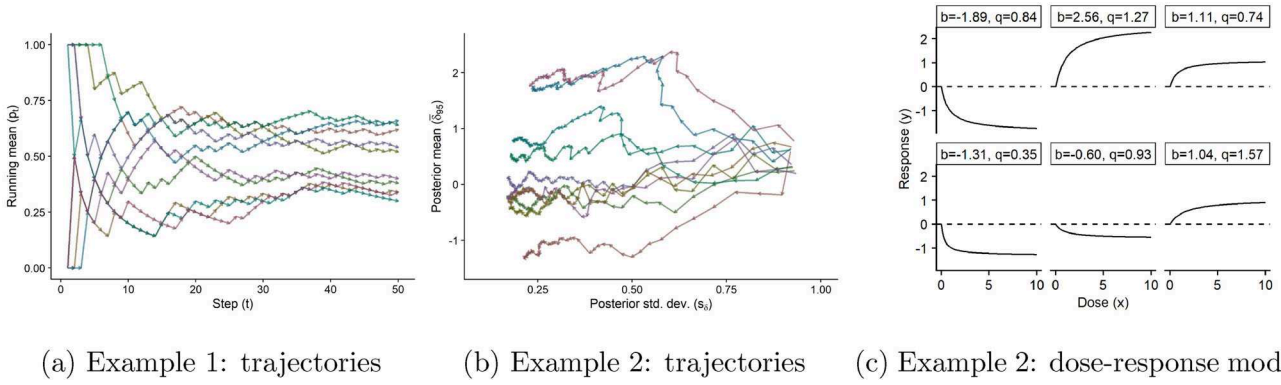
**Example 1: A binary hypothesis.** Consider the decision problem of choosing between  $H_1 : \theta = \theta_1$  and  $H_2 : \theta = \theta_2$ . For instance,  $\theta$  could represent the probability of a clinical response for an experimental therapy. Assume a binary outcome  $Y_t$  with a Bernoulli sampling model  $p(Y_t = 1 | \theta) = \theta$  and a discrete two-point prior  $p(\theta = \theta_1) = p(\theta = \theta_2) = \frac{1}{2}$ .

The possible decisions at any time are  $D_t \in \{0, 1, 2\}$ . Here  $D_t = 0$  indicates continuation,  $D_t = 1$  indicates to terminate the trial and report  $H_1$  ( $\theta = \theta_1$ ), and  $D_t = 2$  means terminate and report  $H_2$  ( $\theta = \theta_2$ ). The utility function includes a (fixed) sampling cost  $c$  for each cohort and a final cost  $K > 0$  for reporting the wrong hypothesis. The utility function is

$$u(H_T, D_T, \theta) = -cT - K \mathbb{I}(\theta \neq \theta_{D_T}). \quad (3)$$

The relevant history  $H_t$  can be represented using the summary statistic  $S_t = (t, \sum_{k \leq t} Y_k / t)$ , since this statistic is *sufficient* for the posterior of  $\theta$ . The implementations of simulation-based BSD and RL use this summary statistic. The problem parameters are fixed as  $c = 1$  and  $K = 100$ . Example trajectories of  $S_t$  assuming no stopping are shown in Figure 2(a).

**Example 2: A dose-finding study.** This example is a stylized version of the ASTIN trial (Grieve and Krams 2005). The trial aims to find the optimal dose for a drug from a set of candidate doses  $\mathbb{X} = \{x_0, \dots, x_G\}$  where  $x_0 = 0$  stands for placebo. At each time  $t$ , a dose  $X_t \in \mathbb{X}$  is assigned to the next patient (cohort), and an efficacy outcome  $Y_t$  is observed. The aim is to learn about the dose–response curve  $f(X_t) = \mathbb{E}(Y_t | X_t)$ , and



**Figure 2.** Data from the two sequential stopping examples. (a) and (b) are forward simulations assuming no stopping; (c) shows the implied dose–response curves for different random draws of the prior.

more specifically, to find the dose  $x_{95}$  (the ED95) that achieves 95% of the maximum possible improvement over placebo. Let  $\delta_g = f(x_g) - f(x_0)$  be the advantage over the placebo at dose  $x_g$ . We set up a nonlinear regression  $Y_t = f(X_t | \theta) + \epsilon_t$  with  $\epsilon_t \sim N(0, \sigma^2)$  using a dose–response function

$$f(x | \theta) = a + b \frac{x^r}{(q^r + x^r)} \quad (4)$$

with  $\theta = (a, b, q, r)$  and a prior  $p(\theta) = N(\theta_0, \text{diag}(\lambda_0^2))$ . In the PK/PD (pharmacokinetics/pharmacodynamics) literature model (4) is known as the  $E_{\max}$  model (Meibohm and Derendorf 1997).

Similar to Example 1,  $D_t \in \{0, 1, 2\}$  with  $D_t = 0$  indicating continuation,  $D_t = 1$  indicating stopping the trial and recommending no further investigation of the experimental therapy, and  $D_t = 2$  indicating stopping and recommending for a follow-up trial set up as a pivotal trial to test the null hypothesis  $H_0 : \delta_{95} = 0$ . If continuing the trial, the next assigned dose is  $X_{t+1} = \min\{X_t + \xi, \hat{x}_{95,t}\}$  where  $\hat{x}_{95,t}$  is the latest estimate of the ED95, and  $\xi$  is a maximum allowable dose escalation between cohorts. If requiring a pivotal trial,  $N_{\alpha,\beta}$  patients are assigned to the dose  $\hat{x}_{95,T}$ , with  $N_{\alpha,\beta}$  computed from the observed data to achieve a desired power  $1 - \beta$  at a certain alternative  $H_1 : \theta = \theta_1$  for test of size  $\alpha$ . Details are given in Appendix A.

The utility function includes a patient recruitment cost of  $c$  and a prize  $K > 0$  if the null hypothesis  $H_0(\delta_{95} = 0)$  is rejected in the pivotal trial (meaning the agent found evidence of an effective drug). Denote  $\Delta_R = \Pr(\text{reject } H_0 \text{ in the 2nd trial} | H_T)$ . At the stopping time  $T = \min_t\{D_t \neq 0\}$ , utility is calculated as

$$u(H_T, D_T, \theta) = \begin{cases} -cT & \text{if } D_T = 1 \\ -cT + \{-cN_{\alpha,\beta}(H_T) + K\Delta_R(H_T)\} & \text{if } D_T = 2 \end{cases} \quad (5)$$

In our implementation we fix  $a = 0$ ,  $r = 1$ ,  $\sigma = 1$  in (4) and  $c = 1$ ,  $K = 100$  in (5), and  $\xi = 1$ , leaving the unknown parameters  $\theta = (b, q)$ , including the maximum effect  $b$  and the location of the  $x_{50}$ . The prior of  $\theta$  has mean  $\theta_0 = (1/2, 1)$  with variances  $\lambda_0 = (1, 1)$  and we add the constraint  $q \geq 0.1$ .

The summary statistic is  $S_t = (\bar{\delta}_{95}, s_8)$  where  $\bar{\delta}_{95}$  and  $s_8$  are posterior mean and standard deviation of  $\delta_{95}$ . Figure 2(b) shows

examples of trajectories of these summaries until some maximum time horizon  $T$  (i.e., assuming no stopping). The trajectories are created by sampling  $\theta$  from the prior, assigning doses as described, and sampling responses using (4). Figure 2(c) shows examples of the implied dose–response curve  $f(x | \theta)$  for different prior draws. Notice that the chosen summaries do not capture the full posterior. The statistic  $S_t$  is not a sufficient statistic for the posterior. However, as shown in Appendix A,  $N_{\alpha,\beta}$  and  $\Delta_R$ , and therefore the utility, depend on the data only through  $S_t$ .

### 3. Simulation-based Bayesian Sequential Design

From the expected utility definition in (1), one immediately deduces that

$$U(H_t, D_t = d) = E\{U(H_{t+1}(D_t = d), D^*(H_{t+1}(D_t = d))) | H_t\}, \quad (6)$$

with the expectation being with respect to future data and  $\theta$ , and substituting optimal choices for future decisions,  $s > t$ . In words, for a rational agent taking optimal actions, the expected utility given history  $H_t$  must be the same as the expected utility in the next step. Thus, one can (theoretically) deduce  $D_t^*$  from knowing the best actions for all possible future histories by implementing *backward induction* starting with  $T$ . Equation (6) is known as the *Bellman equation* (Bellman 1966).

Enumerating all histories is computationally intractable in most realistic scenarios, rendering backward induction usually infeasible, except in some special setups (Berry and Ho 1988; Christen and Nakamura 2003). Simulation-based BSD comes to help: instead of enumerating all possible histories, we compute approximations using some simulated trajectories. The version presented here follows Müller et al. (2007). Similar schemes are developed in Brockwell and Kadane (2003), Kadane and Vlachos (2002), and Carlin, Kadane, and Gelfand (1998). We use two strategies: first, we represent history  $H_t$  through a (low-dimensional) summary statistic  $S_t$ , as already hinted in Section 2 when we proposed the posterior moments of the ED95 response for Example 2. The second—and closely related—strategy is to restrict  $D_t$  to depend on  $H_t$  only indirectly through  $S_t$ . Two instances of this approach are discussed below and used to solve Examples 1 and 2.

### 3.1. Constrained Backward Induction

Constrained backward induction is an algorithm consisting of three simple steps. The first step is *forward simulation*. Our implementation here uses the assumption that the sequential nature is limited to sequential stopping, so trajectories can be generated assuming no stopping independently from decisions. Throughout we use  $D_t = 0$  to denote continuation. Other actions,  $D_t \neq 0$ , indicate stopping the study and choice of a terminal decision. The second step is constrained *backward induction*, which implements (6) using decisions restricted to depend on the history  $H_t$  indirectly only through  $S_t$ . The third step simply keeps track of the best action and iterates until convergence. We first briefly explain these steps and then provide an illustration of their application in [Example 1](#) and additional implementation considerations.

Step 1. *Forward simulation*: Simulate many trajectories, say  $M$ , until some maximum number of steps  $T_{\max}$  (e.g., cohorts in a trial). To do this, each  $m = 1, \dots, M$  corresponds to a different prior draw  $\theta^{(m)} \stackrel{\text{iid}}{\sim} p(\theta^{(m)})$  and samples  $Y_t^{(m)} \stackrel{\text{iid}}{\sim} p(Y_t^{(m)} \mid \theta^{(m)})$ ,  $t = 1, \dots, T_{\max}$ . For each  $m$  and  $t$ , we evaluate and record the summary statistic  $S_t^{(m)}$  discretized over a grid.

Step 2. *Backward induction*: For each possible decision  $d$  and each grid value  $S = j$ , the algorithm approximates  $\hat{U}(S, d) \approx U(S, d)$  using the forward simulation and Bellman equation as follows. Denote with  $A_j = \{(m, t_m) \mid S_{t_m}^{(m)} = j\}$  the set of forward simulations that fall within grid cell  $j$ . Then,

$$\begin{aligned} \hat{U}(S = j, d) &= \begin{cases} \frac{1}{|A_j|} \sum_{(m, t_m) \in A_j} \hat{U}(S_{t_m+1}^{(m)}, D^*(S_{t_m+1}^{(m)})) & d = 0 \\ \frac{1}{|A_j|} \sum_{(m, t_m) \in A_j} u(S_{t_m}^{(m)}, D_{t_m} = d, \theta^{(m)}) & d \neq 0. \end{cases} \quad (7) \end{aligned}$$

The evaluation under  $d = 0$  requires the optimal actions  $D_{t+1}^*(S_{t_m+1})$ . We use an initial guess (see below), which is then iteratively updated (see next).

Step 3. *Iteration*: Update the table  $D^*(S) = \arg \max_d \hat{U}(S, d)$  after step 2.

Repeat Steps 2 and 3 until updating in step 3 requires no (or below a minimum number of) changes of the tabulated  $D^*(S)$ .

[Figure 3\(a\)](#) shows the estimated utility function  $\hat{U}(S, d)$  in [Example 1](#) using  $M = 1000$ ,  $T_{\max} = 50$  and 100 grid values for the running average  $p_t = \sum_{k=1}^t Y_k/t$  in  $S = (t, p_t)$ . Optimal actions  $D_t^*(S)$  are shown in [Figure 3\(b\)](#). The numerical uncertainty due to the Monte Carlo evaluation of the expectations is visible. If desired, one could reduce it by appropriate smoothing (MacDonald, Ranjan, and Chipman 2015). One can verify, however, that the estimates are a close approximation to the analytic solution which is available in this case (Müller et al. 2007).

We explain Step 2 by example. Consider [Figure 3\(b\)](#) and assume, for example, that we need the posterior expected utility of  $S = (t, p_t) = (20, 0.25)$ . In this stylized representation, only three simulations,  $A = \{m_1, m_2, m_3\}$  pass through this grid cell. In this case,  $t_m = t = 20$  for the three trajectories since  $t$  is part of the summary statistic. We evaluate  $\hat{U}(S, d = 1)$  and  $\hat{U}(S, d = 2)$  as averages  $\frac{1}{3} \sum_{i \in A} u(S_{20}^{(m)}, d, \theta^{(m)})$ . For  $\hat{U}(S, d = 0)$ , we first

determine the grid cells in the next period for each of the three trajectories.  $S_{t_m+1}^{(m)} = (21, p_{21}^{(m)})$ . We then look up the optimal decisions  $D^*(S_{21}, p_{21}^{(m)})$  (using in this case  $t_m + 1 = 21$ ) and average  $\frac{1}{3} \sum \hat{U}(21, p_{21}^{(m)}, D^*(21, p_{21}^{(m)}))$ , as in (7).

Constrained backward induction requires iterative updates of  $D^*(S)$  and their values. The procedure starts with arbitrary initial values for  $D^*(S)$ , recorded on a grid over  $S$ . For example, a possible initialization is  $D^*(S) = \max_{d \neq 0} \hat{U}(S, d)$ , maximizing over all actions that do not involve continuation. With such initial values,  $\hat{U}(S, d)$  can be evaluated over the entire grid. Then, for updating the optimal actions  $D^*(S)$  one should best start from grid values that are associated with the time horizon  $T$ , or at least high  $t$ . This is particularly easy when  $t$  is an explicit part of  $S_t$ , as in [Example 1](#) with  $S_t = (t, p_t)$ . Another typical example arises in [Example 2](#) with  $S_t = (\mu_t, \sigma_t)$ , the mean and standard deviation of some quantity of interest. For large  $t$  we expect small  $\sigma_t$ , making it advisable to start updating in each iteration with the grid cells corresponding to smallest  $\sigma_t$ . We iterate until no more (or few) changes happen.

The algorithm can be understood as an implementation of (6).

Consider  $f(S, d)$  as an arbitrary function over pairs  $(S, d)$  and the function operator  $\mathcal{P}f$  defined as  $(\mathcal{P}f)(S, d) = \max_{d'} \mathbb{E}[f(S', d') \mid S]$  where  $S'$  is the summary statistic resulting from sampling one more data point  $Y$  from the unknown  $\theta$  and recompute  $S'$  from  $S$ . Then Bellman equation (6) can be written as  $U = \mathcal{P}U$ . In other words, expected utility under the optimal decision  $D^*$  is a fixed point of the operator  $\mathcal{P}$ . Constrained backward induction attempts to find an approximate solution to the fixed-point equation. The same principle motivates the Q-learning algorithm in RL (Watkins and Dayan 1992) (see [Section 4](#)). Backward induction is also closely related to the *value iteration* algorithm for Markov decision processes (Sutton and Barto 2018), which relies on exact knowledge of the state transition function.

### 3.2. Sequential Design with Decision Boundaries

Inspection of [Figure 3\(b\)](#) suggests an attractive alternative algorithm. Notice the decision boundaries on  $S = (t, p_t)$  that trace a funnel with an upper boundary  $\omega_2(t)$  separating  $D^* = 2$  from  $D^* = 0$ , and a lower boundary  $\omega_1(t)$  separating  $D^* = 0$  versus  $D^* = 1$ .

Recognizing such boundaries suggests an alternative approach based on searching for optimal boundaries in a suitable family  $\{\omega_{\phi,1}, \omega_{\phi,2} \mid \phi \in \Phi\}$ .

This approach turns the sequential decision problem of finding optimal  $D^*(S)$  into a nonsequential problem of finding an optimal  $\phi^* \in \Phi$ . This method is used, for example, in Rossell, Müller, and Rosner (2007).

In [Example 1](#), we could use

$$\omega_1(t) = \frac{\phi\sqrt{t-1}}{\sqrt{T-1}}, \quad \omega_2(t) = 1 - \frac{(1-\phi)\sqrt{t-1}}{\sqrt{T-1}}$$

using a single tuning parameter  $\phi \in (0, 1)$ . Both functions are linear in  $\sqrt{t-1}$ , and mimic the funnel shape seen in

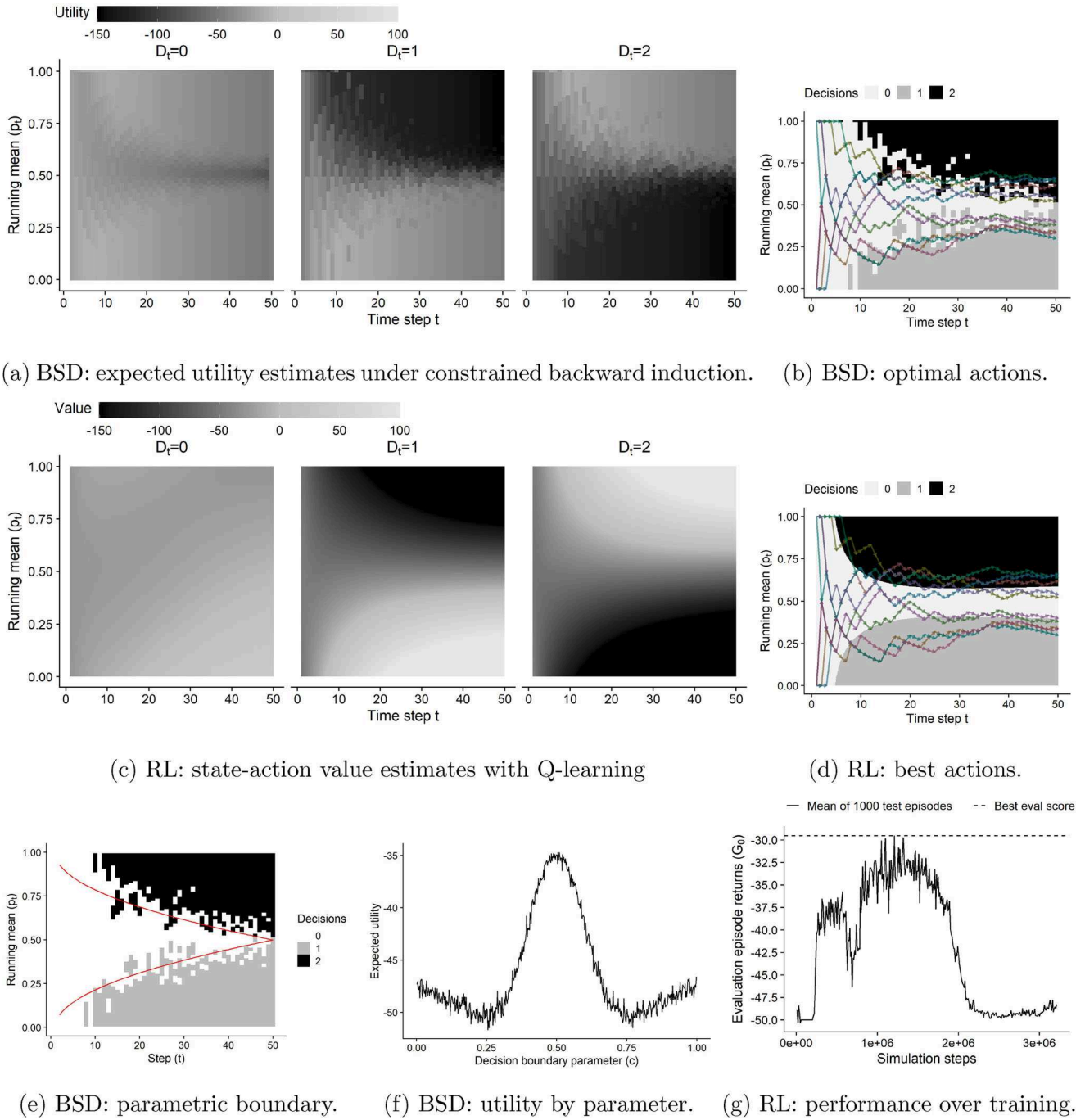


Figure 3. Example 1. Comparison of decision boundaries and fitted value functions/utilities.

Figure 3(b). The decision rules implied by these boundaries is

$$D_\phi(S_t) = \begin{cases} 1 & \text{if } p_t < \omega_1(t) \\ 2 & \text{if } p_t > \omega_2(t) \\ 0 & \text{otherwise.} \end{cases}$$

Here, the additional subscript  $\phi$  in  $D_\phi(\cdot)$  indicates that the decision follows the rule implied by decision boundaries  $\omega_j(t; \phi)$ . Note that  $\omega_1(1) = 1$  and  $\omega_2(1) = 0$ , ensuring continuation at  $t = 1$ .

For a given  $\phi$ , the forward simulations are used to evaluate expected utilities under the policy  $D_\phi$ . Let  $T_\phi^{(m)} = \min\{t :$

$D_\phi(S_t^{(m)}) \neq 0\}$  denote the stopping time under  $D_\phi$ . Then the expected utility under policy  $D_\phi$  is

$$U(\phi) = E\{u(S_{T_\phi}, D_\phi(S_{T_\phi}), \theta)\} \quad (8)$$

where the expectation is with respect to data  $Y_T$  and  $\theta$ . It is approximated as an average over all Monte Carlo simulations, stopping each simulation at  $T_\phi^{(m)}$ , as determined by the parametric decision boundaries,

$$\hat{U}(\phi) = (1/M) \sum_{m=1}^M u(S_{T_\phi^{(m)}}^{(m)}, D_\phi(S_{T_\phi^{(m)}}^{(m)}), \theta^{(m)}). \quad (9)$$



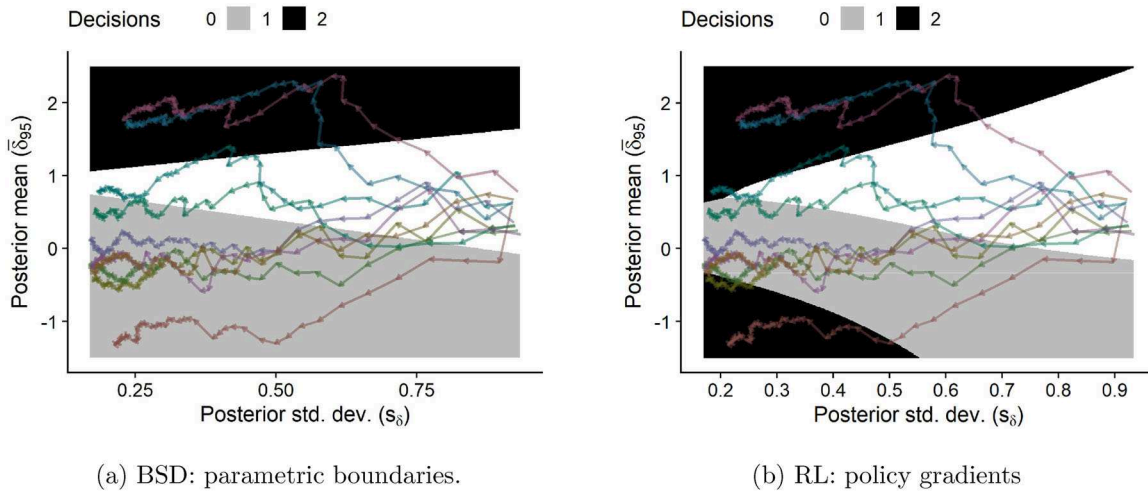


Figure 4. Optimal decisions in Example 2. Comparison of fitted decision boundaries.

Optimizing  $\hat{U}(\phi)$  w.r.t.  $\phi$  we find the optimal decision boundaries  $\phi^* = \arg \max \hat{U}(\phi)$ . As long as the nature of the sequential decision is restricted to sequential stopping, the same set of Monte Carlo simulations can be used to evaluate all  $\phi$ , using different truncation to evaluate  $\hat{U}(\phi)$ . In general, a separate set of forward simulations for each  $\phi$ , or other simplifying assumptions might be required.

Figure 3(e) shows the decision boundaries for the best parameter estimated at  $\phi^* = 0.503$  in Example 1. The estimated values for  $\hat{U}(\phi)$  are in Figure 3(f). In Figure 3(e), the boundaries using constrained backward induction are overlaid in the image for comparison. The decision boundaries trace the optimal decisions under the backward induction well. The differences in expected utility close to the decision boundary are likely very small, leaving minor variations in the decision boundary negligible.

The same approach is applied to the (slightly more complex) Example 2.

Recall the form of the summary statistics  $S = (s_\delta, \bar{\delta})$ , the posterior standard deviation and mean of the ED95 effect. We use the boundaries

$$\omega_1(S) = -b_1 s_\delta + c, \quad \omega_2(S) = b_2 s_\delta + c,$$

parameterized by  $\phi = (b_1, b_2, c)$ . The implied decision rules are

$$D_\phi(S) = \begin{cases} 1 & \text{if } \bar{\delta} < \omega_1(s_\delta) \\ 2 & \text{if } \bar{\delta} > \omega_2(s_\delta) \\ 0 & \text{otherwise.} \end{cases}$$

The results are in Figure 4(a). Again, the sequential decision problem is reduced to the optimization problem of finding the optimal  $\phi$  in (9). Since now  $\phi \in \mathbb{R}^3$  the evaluation of  $\hat{U}$  requires a three-dimensional grid. To borrow strength from Monte Carlo evaluations of (9) across neighboring grid points for  $\phi$  we proceed as follows. We evaluate  $\hat{U}(\phi)$  on a coarse  $10 \times 10 \times 10$  grid, and then fit a quadratic response surface (as a function of  $\phi$ ) to these Monte Carlo estimates. The optimal decision  $\phi^*$  is the maximizer of the quadratic fit. We find  $\phi^* = (b_1^*, b_2^*, c^*) = (1.572, 1.200, 0.515)$ . Instead of evaluating  $\hat{U}$  on a regular grid over  $\phi$ , one could alternatively select a random number of design points (in  $\phi$ ).

The use of parametric boundaries is closely related to the notion of function approximation and the method of policy gradients in RL, which will be described next.

## 4. Reinforcement Learning

The basic setup in RL is usually framed in terms of Markov decision process (MDP) (Puterman 2014). The Markov property ensures that optimal decisions depend only on the most recently observed state, enabling practicable algorithms. In this section we first describe MDPs and how a sequential design problem can be adapted to fit in this framework. Next, we discuss two algorithms, Q-learning (Watkins and Dayan 1992) and policy gradients (Grondman et al. 2012), implemented in Examples 1 and 2, respectively. Both methods are implemented using neural networks. Throughout this section, the summary statistics  $S_t$  are referred to as *states*, in keeping with the common terminology in the RL literature.

### 4.1. Markov Decision Processes and Partial Observability

The Markov property for a decision process is defined by the conditions

$$p(S_{t+1} | H_t, D_t) = p(S_{t+1} | S_t, D_t) \\ \text{and } p(R_t | H_t, D_t) = p(R_t | S_t, D_t);$$

that is, the next state  $S_{t+1}$  and the reward  $R_t$  depend on the history only indirectly through the current state and action. When the condition holds, the decision process is called an MDP. For MDPs, the optimal policy is only a function of the latest state  $S_t$  and not of the entire history  $H_t$  (Puterman 2014). Many RL algorithms assume the Markov property. However, many sequential decision problems are more naturally characterized as *partially observable MDP* (POMDP) that satisfy the Markov property only conditional on some  $\theta$  that is generated at the beginning of each episode. Such problems have been studied in the RL literature under the name of Hidden-Parameter MDP (HiMDP) (Doshi-Velez and Konidaris 2016).

There is a standard—Bayesian motivated—way to cast any POMDP as an MDP using so-called *belief states* (Cassandra,



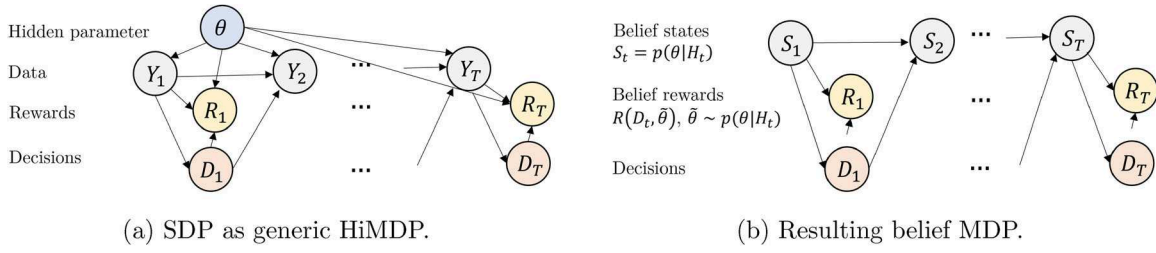


Figure 5. Belief MDP for the SDP.

Littman, and Zhang 1997). Belief states are obtained by including the posterior distribution of unobserved parameters as a part of the state. With a slight abuse of notation, we may write the belief states as  $S_t = p(\theta | H_t)$ . While the belief state is, in general, a function, it can often be represented as a vector when the posterior admits a finite (sufficient) summary statistic. The reward distribution can also be written in terms of such belief states as

$$p(R | S_t, D_t) = \int_{\theta} p(R | D_t, \theta) dp(\theta | H_t). \quad (10)$$

See Figure 5 for a graphical representation of an HiMDP and the resulting belief MDP.

We implement this approach for Example 1. The reward is chosen to match the definition of utility. It suffices to define it in terms of  $\theta$  (and let the posterior take care of the rest, using (10)). We use

$$R(d, \theta) = -c \mathbb{I}(d = 0) - K \mathbb{I}(\theta_d \neq \theta, d \neq 0), \quad (11)$$

as in (3). Next we introduce the belief states. Recall the notation from Example 1. We have  $p(\theta | S_t) = \text{Bin}(p_t, p_t(1 - p_t)/t)$ . The summary statistic  $S_t = (t, p_t)$  is a two-dimensional representation of the belief state.

Considering Example 2, we note that the utility function (5) depends on the state  $S_t$  only, and does not involve  $\theta$ . We define

$$R_{\theta}(d, S_t) = -c_1 \mathbb{I}(d = 0) + (-c_2 N_{\alpha, \beta}(S_t) + K \Delta_R(S_t)) \mathbb{I}(d = 2). \quad (12)$$

While the reward is clearly Markovian, the transition probability is not necessarily Markovian. This is the case because in this example the posterior moments  $S_t$  are not a sufficient statistic. In practice, however, a minor violation of the Markov assumption for the transition distribution does not seem to affect the ability to obtain good policies with standard RL techniques.

#### 4.2. Q-learning

Q-learning (Watkins and Dayan 1992; Murphy 2003; Clifton and Laber 2020) is an RL algorithm that is similar in spirit to the constrained backward induction described in Section 3.1. The starting point is Bellman optimality equation for MDPs (Bellman 1966). Equation (6) for the optimal  $D^*$  and written for MDPs becomes

$$Q^{\pi^*}(s, d) = E\{R_t + \max_{d'} Q^{\pi^*}(S_{t+1}, d') | S_t = s, D_t = d\}, \quad (13)$$

where the expectation is with respect to  $R_t$  and  $S_{t+1}$ . The optimal policy is implicitly defined as the solution to (13).

Q-learning proceeds iteratively following the fixed-point iteration principle. Let  $Q^{(k)}$  be some approximation of  $Q^*$ . We assume that a set of simulated transitions  $\{(s_t, d_t, r_t, s_{t+1})\}_{t=1}^n$  is available. This collection is used like the forward simulations in the earlier discussion. In RL it is known as the “experience replay buffer,” and can be generated using any stochastic policy. And suppose, for the moment, that the state and action spaces are finite discrete, allowing to record  $Q^{(k)}$  in a table. Q-learning is defined by updating  $Q^{(k)}$  as

$$Q^{(k+1)}(s_t, d_t) \leftarrow (1 - \alpha_k) Q^{(k)}(s_t, d_t) + \alpha_k \{r_t + \max_d Q^{(k)}(s_{t+1}, d)\}. \quad (14)$$

Note the moving average nature of the update. The procedure iterates until convergence from a stream of transitions.

Deep Q-networks (DQN) (Mnih et al. 2013) are an extension of Q-learning for continuous states. A neural network is used to represent  $Q(\cdot)$ . Let  $\phi^{(k)}$  denote the parameters of the neural network at iteration  $k$ . Using simulated transitions from the buffer

DQN performs updates

$$\phi^{(k+1)} = \arg \min_{\phi} \sum_{t=1}^n \left( r_t + \max_d Q_{\phi^{(k)}}(S_{t+1}, d) - Q_{\phi}(S_t, d_t) \right)^2. \quad (15)$$

In practice, exact minimization is replaced by a gradient step from mini-batches, together with numerous implementation tricks (Mnih et al. 2013).

We implemented DQN in Example 1 using the Python package `Stable-Baselines3` (Raffin et al. 2021). The experience replay buffer is continuously updated. The algorithm uses a random policy to produce an initial buffer and then adds experience from an  $\epsilon$ -greedy policy, where the current best guess for the optimal policy is chosen with probability  $(1 - \epsilon)$ , and otherwise fully random actions are chosen with probability  $\epsilon$ . Figure 3(c) shows  $\hat{Q}$ , the estimate of  $Q^*$ , for each state and action  $d \in \{0, 1, 2\}$ . Figure 3(d) shows the corresponding optimal actions. Overall, the results are similar to the results with constrained backward induction, but much smoother. Also, notice that the solution under DQN is usually better in terms of expected utility as shown in Figure 3(f), even in (out-of-sample) evaluation episodes. This is likely due to the flexibility and high-dimensional nature of the neural network approximation.

The better performance of RL comes at a price. First is sample efficiency (the number of simulations required by the algorithm to yield a good policy). The best  $\hat{Q}$  is obtained after 2 million sampled transitions. Data efficiency is a known problem in DQN, and in RL in general (Yu 2018). In many real applications investigators cannot afford such a high number of simulation

steps. Another limitation is training instability. In particular, Figure 3(f) illustrates a phenomenon known as *catastrophic forgetting*, which happens when additional training decreases the ability of the agent to perform a previously learned task (Atkinson et al. 2021). This can happen because of the instability that arises from a typical strategy of evaluating performance periodically and keeping track of the best performing policy only. Several improvements over basic DQN have been proposed, with improved performance and efficiency (Hessel et al. 2018).

### 4.3. Policy Gradients

The approach is similar to the use of parametric boundaries discussed before. Policy gradient (PG) approaches start from a parameterization  $\pi_\phi$  of the policy. Again, consider a neural network (NN) with weights  $\phi$ . The goal of a PG method is to maximize the objective

$$\max_{\phi} J(\phi) = E\{G \mid \pi_{\phi}\}. \quad (16)$$

This objective is the analogue to maximizing  $U(\phi)$  in (9), except that here the stochastic policy  $\pi_\phi$  is a probability distribution on decisions  $S_t$ . The main characteristic of PG methods is the use of gradient descent to solve (16).

The evaluation of gradients is based on the PG theorem (Sutton et al. 1999),

$$\nabla_{\phi} J(\phi) = E \left\{ \left[ \sum_{t=1}^T \nabla_{\phi} \log \pi_{\phi}(D_t \mid S_t) \right] \cdot G \mid \pi_{\phi} \right\}, \quad (17)$$

where the total return  $G$  is a function  $G(\tau)$  of the entire trajectory  $\tau = (S_1, D_1, \dots, S_T, D_T)$ . Using gradients, PG methods can optimize over high-dimensional parameter spaces like in neural networks. In practice, estimates of the gradient are known to have huge variance, affecting the optimization. But several implementation tricks exist that improve the stability and reduce the variance. Proximal policy optimization (PPO) (Schulman et al. 2017) incorporates many of these tricks, and is widely used as a default for PG-based methods.

The PG theorem is essentially Leibniz rule for the gradient of  $J(\phi)$ .

With a slight abuse of notation, write  $\pi_{\phi}(\tau)$  for the distribution of  $\tau$  induced by  $\pi_{\phi}$  for the sequential decisions. Then Leibniz rule for the gradient of the integral gives

$$\begin{aligned} \nabla_{\phi} J(\phi) &= \nabla_{\phi} \int \pi_{\phi}(\tau) G(\tau) d\tau = \int \nabla_{\phi} \pi_{\phi}(\tau) \cdot G(\tau) d\tau \\ &= \int (\nabla_{\phi} \log \pi_{\phi}(\tau)) \pi_{\phi}(\tau) \cdot G(\tau) d\tau \\ &= E \{ \nabla_{\phi} \log \pi_{\phi}(\tau) \cdot G(\tau) \}, \end{aligned}$$

where the expectations are with respect to the (stochastic) policy  $\pi_{\phi}$  over  $\tau$ . The log probability in the last expression can be written as a sum of log probabilities, yielding (17).

PPO is implemented in Example 2 using `Stable-Baselines3` (Raffin et al. 2021). The results are shown in Figure 4(b). Not surprisingly, the results are similar to

those obtained earlier using parametric decision boundaries. Interestingly, the figure shows that neural networks do not necessarily extrapolate well to regions with low data. This behavior is noticeable on the lower left corner of the figure, where there *could* be data, but where it is never observed in practice because of the early stopping implied by the boundaries.

## 5. Discussion

We have introduced some of the main features of RL and BSD in the context of two optimal stopping problems. In the context of these examples the two approaches are quite similar, including an almost one-to-one mapping of terminology and notation, as we attempted in Table 1. In general, however, the applicability, especially the practical use of RL is much wider. The restriction of the sequential problems to optimal stopping was only needed for easy application of the BSD solution. In contrast, RL methods are routinely used for a variety of other problems, such as robotics (Tunyasuvunakool et al. 2020) autonomous driving (Sallab et al. 2017; Wurman et al. 2022), and smart building energy management (Yu et al. 2021). The main attraction of BSD is the principled nature of the solution. One can argue from first principles that a rational agent should act as if he or she were optimizing expected utility as in (1). There is a well-defined and coherent propagation of uncertainties. This might be particularly important when the SDP and underlying model are only part of a bigger problem. Overall, we note that the perspective of one method, and corresponding algorithms can be useful for improvements in the respective other method. For example, policy gradients could readily be used to solve BSD if randomized decision rules were used. The latter is usually not considered. On the other hand, hierarchical Bayesian inference models could be used to combine multiple sources of evidence in making sequential decisions under RL, or multiple related problems could be linked in a well-defined manner in a larger encompassing model. For example, clinical trials are never carried out in isolation. Often the same department or group might run multiple trials on the same patient population for the same disease, with obvious opportunities to borrow strength.

## Appendix A. Details in Example 2

We assume a nonlinear regression sampling model

$$Y_t = f(X_t \mid \theta) + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2),$$

and the dose-response curve

$$f(X_t \mid \theta) = a + b \frac{X_t^r}{(q^r + X_t^r)}$$

We fix  $a, r, \sigma^2$ , and put a normal prior on unknown parameters  $\theta = (b, q)$ ,

$$p(\theta) = N(\theta_0, \text{diag}(\lambda_0)).$$

**Sample size calculation.** If at time  $T$ , the decision  $D_T = 2$  indicates stopping and a pivotal trial is conducted to test  $H_0 : \delta_{95} = 0$  versus  $H_1 : \delta_{95} > 0$ . We need to determine the sample size  $N_{\alpha, \beta}$  for the pivotal trial that can achieve desired significance level  $\alpha$  and power

$(1 - \beta)$ , and calculate the posterior predictive probability of a significant outcome,  $\Delta_R = \Pr(\text{reject } H_0 \text{ in the 2nd trial} \mid H_T)$ .

Let  $\bar{\delta}_{95}$  and  $s_\delta$  denote the posterior mean and SD of  $\delta_{95}$ . We calculate power based on  $\delta_{95} = \delta^*$ , where  $\delta_{95}^* = \bar{\delta}_{95} - s_\delta$ .

Now consider a test enrolling  $N_{\alpha,\beta}$  patients, randomizing  $N_{\alpha,\beta}/2$  at  $x = 0$  (placebo) and  $N_{\alpha,\beta}/2$  at the estimated ED95. Assuming  $\text{var}(y_i) = 1$ , we need

$$N_{\alpha,\beta} \geq 4[(q_\alpha + q_\beta)/\delta^*]^2$$

where  $q_\alpha$  is the  $\alpha$  right tail cutoff for the  $N(0, 1)$  and  $\alpha = 5\%$  and  $(1 - \beta) = 80\%$  are the desired significance level and power (i.e.,  $\beta = 0.2$ ).

A significant outcome at the end of the second trial means data in the rejection region. Let  $\bar{y}_1, \bar{y}_0$  denote the sample average of  $N_{\alpha,\beta}/2$  patients each to be enrolled in the two arms of the second trial. Then the rejection region is

$$R = \{(\bar{y}_1 - \bar{y}_0)\sqrt{N_{\alpha,\beta}/4} \geq q_\alpha\}.$$

Let  $\Phi(\cdot)$  denote the standard normal cdf. Then

$$\Delta_R = \Phi\left[\frac{\bar{\delta}\sqrt{N_{\alpha,\beta}/4} - q_\alpha}{\sqrt{1 + \frac{N_{\alpha,\beta}}{4}s_\delta^2}}\right]$$

**Posterior simulation.** We can implement independent posterior simulation:

(i) Generate  $q \sim p(q \mid H_t)$ , using

$$p(q \mid H_t) \propto p(H_t \mid q) \cdot p(q) = \frac{p(H_t \mid b, q)p(b)}{p(b \mid q, H_t)} \cdot p(q)$$

(ii) Then generate  $b$  from the posterior conditional distribution  $b \sim p(b \mid q, H_t)$ . Based on normal linear regression, the conditional posterior is a univariate normal distribution.

## Acknowledgments

The authors gratefully thank Peter Stone and the Learning Agents Research Group (LARG) for helpful discussions and feedback.

## Disclosure Statement

The authors report there are no competing interests to declare

## Funding

Yunshan Duan and Peter Müller are partially funded by the NSF under grant NSF/DMS 1952679.

## ORCID

Mauricio Tec  <https://orcid.org/0000-0002-1853-5842>

## References

Atkinson, C., McCane, B., Szymanski, L., and Robins, A. (2021), "Pseudo-Rehearsal: Achieving Deep Reinforcement Learning without Catastrophic Forgetting," *Neurocomputing*, 428, 291–307. [231]  
Bellman, R. (1966), "Dynamic Programming," *Science*, 153, 34–37. [226,230]

Berger, J. O. (2013), *Statistical Decision Theory and Bayesian Analysis*, New York: Springer. [223,224]  
Berry, D. A., and Ho, C.-H. (1988), "One-Sided Sequential Stopping Boundaries for Clinical Trials: A Decision-Theoretic Approach," *Biometrics*, 44, 219–227. [223,226]  
Brockwell, A. E., and Kadane, J. B. (2003), "A Gridding Method for Bayesian Sequential Decision Problems," *Journal of Computational and Graphical Statistics*, 12, 566–584. [226]  
Carlin, B. P., Kadane, J. B., and Gelfand, A. E. (1998), "Approaches for Optimal Sequential Decision Analysis in Clinical Trials," *Biometrics*, 54, 964–975. [226]  
Cassandra, A., Littman, M. L., and Zhang, N. L. (1997), "Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes," in *Proceedings of the Thirteenth conference on Uncertainty in Artificial Intelligence*, pp. 54–61. [230]  
Christen, J. A., and Nakamura, M. (2003), "Sequential Stopping Rules for Species Accumulation," *Journal of Agricultural, Biological, and Environmental Statistics*, 8, 184–195. [223,226]  
Clifton, J., and Laber, E. (2020), "Q-learning: Theory and Applications," *Annual Review of Statistics and its Application*, 7, 279–301. [230]  
DeGroot, M. (2004), *Optimal Statistical Decisions*, New York: Wiley-Interscience. [223,224]  
Doshi-Velez, F., and Konidaris, G. (2016), "Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations," in *IJCAI: Proceedings of the Conference* (Vol. 2016), pp. 1432. [229]  
Gaon, M., and Brafman, R. (2020), "Reinforcement Learning with Non-Markovian Rewards," in *Thirty-fourth AAAI Conference on Artificial Intelligence*. [223]  
Grieve, A. P., and Krams, M. (2005), "ASTIN: A Bayesian Adaptive Dose-Response Trial in Acute Stroke," *Clinical Trials*, 2, 340–351. [225]  
Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012), "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42, 1291–1307. [229]  
Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018), "Rainbow: Combining Improvements in Deep Reinforcement Learning," in *Thirty-second AAAI Conference on Artificial Intelligence*. [231]  
Kadane, J. B., and Vlachos, P. K. (2002), "Hybrid Methods for Calculating Optimal Few-Stage Sequential Strategies: Data Monitoring for a Clinical Trial," *Statistics and Computing*, 12, 147–152. [226]  
MacDonald, B., Ranjan, P., and Chipman, H. (2015), "GPfit: An R Package for Fitting a Gaussian Process Model to Deterministic Simulator Outputs," *Journal of Statistical Software*, 64, 1–23. [227]  
Meibohm, B., and Derendorf, H. (1997), "Basic Concepts of Pharmacokinetic/Pharmacodynamic (pk/pd) Modelling," *International Journal of Clinical Pharmacology and Therapeutics*, 35, 401–413. [226]  
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013), "Playing Atari with Deep Reinforcement Learning," arXiv preprint 1312.5602. [224,230]  
Müller, P., Berry, D. A., Grieve, A. P., Smith, M., and Krams, M. (2007), "Simulation-based Sequential Bayesian Design," *Journal of Statistical Planning and Inference*, 137, 3140–3150. [223,226,227]  
Murphy, S. A. (2003), "Optimal Dynamic Treatment Regimes," *Journal of the Royal Statistical Society, Series B*, 65, 331–355. [223,230]  
Murphy, S. A., Osolin, D. W., Rush, A. J., and Zhu, J. (2007), "Methodological Challenges in Constructing Effective Treatment Sequences for Chronic Psychiatric Disorders," *Neuropsychopharmacology*, 32, 257–262. [223]  
Parmigiani, G., and Inoue, L. (2009), *Decision Theory: Principles and Approaches*, Chichester: Wiley. [223]  
Puterman, M. L. (2014), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Hoboken, NJ: Wiley. [224,229]  
Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021), "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, 22, 1–8. [230,231]  
Robins, J. M. (1997), "Causal Inference from Complex Longitudinal Data," in *Latent Variable Modeling and Applications to Causality*, ed. M. Berkane, pp. 69–117, New York: Springer. [224]  
Rossell, D., Müller, P., and Rosner, G. (2007), "Screening Designs for Drug Development," *Biostatistics*, 8, 595–608. [223,227]

- Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017), “Deep Reinforcement Learning Framework for Autonomous Driving,” *Electronic Imaging*, 2017, 70–76. [224,231]
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017), “Proximal Policy Optimization Algorithms,” arXiv preprint 1707.06347. [231]
- Shen, W., and Huan, X. (2021), “Bayesian Sequential Optimal Experimental Design for Nonlinear Models Using Policy Gradient Reinforcement Learning,” arXiv preprint 2110.15335. [223]
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018), “A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and go through Self-Play,” *Science*, 362, 1140–1144. [224]
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014), “Deterministic Policy Gradient Algorithms,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pp. 387–395. [225]
- Sutton, R. S. and Barto, A. G. (2018), *Reinforcement Learning: An Introduction* (Vol. 5), Cambridge, MA: MIT Press, p. 31. [227]
- Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999), “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” in *Advances in Neural Information Processing Systems* (Vol. 12), eds. S. Solla, T. Leen, and K. Müller, MIT Press. [231]
- Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. (2020), “dm\_control: Software and Tasks for Continuous Control,” *Software Impacts*, 6, 100022. [224,231]
- Watkins, C. J., and Dayan, P. (1992), “Q-learning,” *Machine Learning*, 8, 279–292. [227,229,230]
- Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H. C., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M., and Kitano, H. (2022), “Outracing Champion Gran Turismo Drivers with Deep Reinforcement Learning,” *Nature*, 602, 223–228. [224,231]
- Yu, L., Qin, S., Zhang, M., Shen, C., Jiang, T., and Guan, X. (2021), “A Review of Deep Reinforcement Learning for Smart Building Energy Management,” *IEEE Internet of Things Journal*, 8, 12046–12063. [231]
- Yu, Y. (2018), “Towards Sample Efficient Reinforcement Learning,” in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. [230]
- Zhao, Y., Kosorok, M. R., and Zeng, D. (2009), “Reinforcement Learning Design for Cancer Clinical Trials,” *Statistics in Medicine*, 28, 3294–3315. [223]