# OPTIMAL LEARNING: COMPUTATIONAL PROCEDURES FOR BAYES-ADAPTIVE MARKOV DECISION PROCESSES

A Dissertation presented

by

MICHAEL O'GORDON DUFF

Submitted to the Graduate School of the

University of Massachusetts Amherst in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2002

Department of Computer Science

# OPTIMAL LEARNING: COMPUTATIONAL PROCEDURES FOR BAYES-ADAPTIVE MARKOV DECISION PROCESSES

A Dissertation Presented

by

MICHAEL O'GORDON DUFF

# ACKNOWLEDGMENTS

I wish to thank the members of my committee: Professors Andrew Barto, Weibo Gong, Paul Utgoff, and Shlomo Zilberstein. In particular, I would like to thank Andy for his sound editorial judgment and good ideas.

# ABSTRACT

## OPTIMAL LEARNING: COMPUTATIONAL PROCEDURES FOR BAYES-ADAPTIVE MARKOV DECISION PROCESSES

FEBRUARY 2002

MICHAEL O'GORDON DUFF

B.S., UNIVERSITY OF ARIZONA

M.S., UNIVERSITY OF ARIZONA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew Barto

This dissertation considers a particular aspect of sequential decision making under uncertainty in which, at each stage, a decision-making agent operating in an uncertain world takes an action that elicits a reinforcement signal and causes the state of the world to change. *Optimal learning* is a pattern of behavior that yields the highest expected total reward over the entire duration of an agent's interaction with its uncertain world. The problem of determining an optimal learning strategy is a sort of meta-problem, with optimality defined with respect to a distribution of environments that the agent is likely to encounter. Given this prior uncertainty over possible environments, the optimal-learning agent must collect and use information in an intelligent way, balancing greedy exploitation of certainty-equivalent world models with exploratory actions aimed at discerning the true state of nature.

My approach to approximating optimal learning strategies retains the full model of the sequential decision process that, in incorporating a Bayesian model for evolving uncertainty about unknown process parameters, takes the form of a Markov decision process defined over a set of "hyperstates" whose cardinality grows exponentially with the planning horizon.

I develop computational procedures that retain the full Bayesian formulation, but sidestep intractability by utilizing techniques from reinforcement learning theory (specifically, Monte-Carlo simulation and the adoption of parameterized function approximators). By pursuing an approach that is grounded in a complete Bayesian world model, I develop algorithms that produce policies that exhibit performance gains over simple heuristics. Moreover, in contrast to many heuristics, the justification or legitimacy of the policies follows directly from the fact that they are clearly motivated by a complete characterization of the underlying decision problem to be solved.

This dissertation's contributions include a reinforcement learning algorithm for estimating Gittins indices for multi-armed bandit problems, a Monte-Carlo gradient-based algorithm for approximating solutions to general problems of optimal learning, a gradient-based scheme for improving optimal learning policies instantiated as finite-state stochastic automata, and an investigation of diffusion processes as analytical models for evolving uncertainty.

# PREFACE

In broad terms, this dissertation is about sequential decision making under uncertainty. At each stage, a decision-making agent operating in an uncertain world takes an action that elicits a reinforcement signal (from the agent's environment or from the agent itself) and causes the state of the world (or agent) to change. The agent's goal is to maximize the total reward it derives over its entire duration of operation—an interval that may require the agent to strike a balance between two sometimes conflicting impulses: (1) greedy exploitation of its current world model, and (2) exploration of its world to gain information that can refine the world model and improve the agent's policy.

Over the years, a number of researchers have formulated this problem mathematically— "adaptive control processes," "dual control," "value of information," and "optimal learning" all address essentially the same issue and share a basic Bayesian framework that is well-suited for modeling the role of information and for defining what a solution is, but classical procedures for computing policies that optimally balance exploitation with exploration are intractable in general, and have only been able to address problems that have a very small number of physical states and short planning horizons. This dissertation proposes computational procedures that retain the full Bayesian formulation, but sidestep intractability by utilizing state-of-the-art reinforcement learning techniques that employ Monte-Carlo simulation and parameterized function approximators.

Historically, one could view the problem of optimal learning as flowing naturally out of an immense body of work (Figure 1) that considers adaptive control processes from a decision-theoretic perspective, whose origins may be traced back to the sequential analysis of Wald (1947).

The study of adaptive control processes is concerned with systems whose behavior depends upon a parameter whose value is unknown. Methodologies for analyzing these systems can be dichotomized (Kumar, 1985) into non-Bayesian (non-parametric) approaches, exemplified by Robbins-Monro schemes (Robbins, 1952) that focus upon asymptotic performance, and Bayesian (parametric) schemes that impose more structure upon the problem (a prior over problem instances). Bayesian approaches have a clearly-defined notion of optimality, and may be viewed as being somewhat more ambitious than non-parametric approaches. This dissertation's perspective is most clearly allied with that of students of Ron Howard's in the early 1960's. Parallel to theory developed for processes with uncertain physical state (Drake, 1962), Howard and his students (Cozzolino, Gonzales-Zubieta, & Miller, 1965; Martin, 1967; Silver, 1963) considered Bayesian control of Markov chains with uncertainty in their transition probabilities, though they encountered computational barriers that limited their approaches to problems having a very small number of physical states and short planning horizons. One could identify the approach pursued in this dissertation with this work of Howard's students, reconsidered in light of reinforcement learning theory's successful application to Markov decision processes with large numbers of states.[1]

Within computer science and artificial intelligence, optimal learning relates to considerations of uncertain knowledge and reasoning, and decision making under uncertainty. Decision-theoretic planning (*e.g.*, Dean & Wellman, 1991) attempts to

[1]For example, perhaps the most famous application of reinforcement learning theory techniques has been to the development of a world-class computer backgammon player (Tesauro, 1994). The game of backgammon can be modeled as a Markov decision process with on the order of $10^{20}$ states.

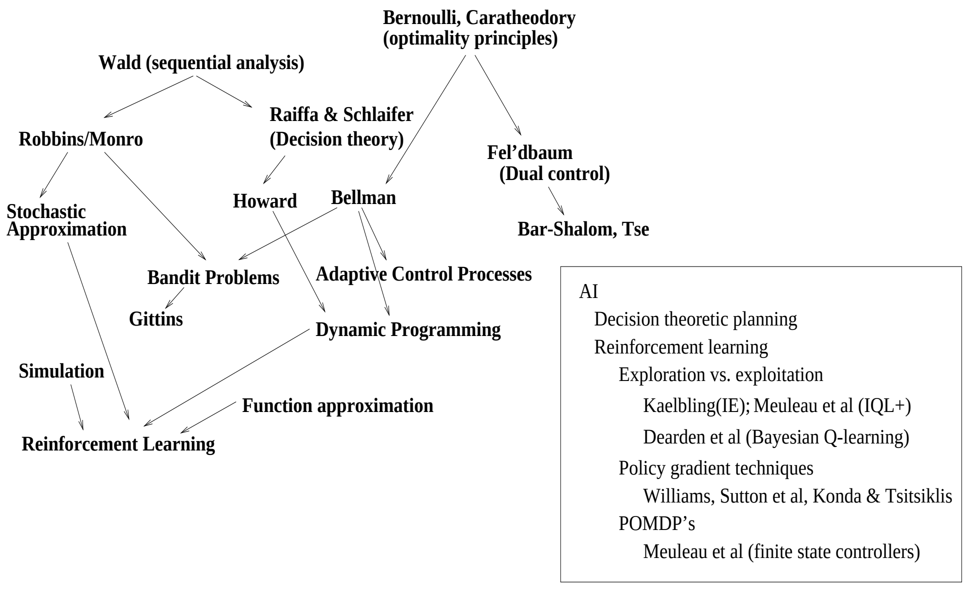


Figure 1: Context of the optimal learning problem. This dissertation adopts a Bayesian framework for modeling adaptive control processes and utilizes techniques from reinforcement learning theory to maintain computational tractability.

import concepts from classical decision and utility theory to the domain of abstract agents operating in uncertain environments. Within reinforcement learning (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998), which itself lies at the intersection of control theory, operations research, applied probability, artificial intelligence, and animal learning, a solution to the problem of optimal learning essentially dictates how an agent should explore. Implicitly, this dissertation seeks approximate solutions to what researchers in the field of reinforcement learning sometimes refer to as the "exploration versus exploitation" problem. My approach is to retain the full model of the sequential decision process, which incorporates a Bayesian model for evolving uncertainty about unknown process parameters. The sequential Bayesian decision process is a Markov decision process with, in general, an enormous number of states (or "hyperstates," which are composed of the physical Markov chain state together with the "information state," which is defined by parameters that characterize evolving uncertainty in the underlying process). In this dissertation, I intend to show how one can approximate optimal policies for this Markov decision process by applying techniques from reinforcement learning theory. One way of thinking about the computational procedures that I later propose is that they perform an offline computation of an online, adaptive machine. One may regard the process of approximating an optimal policy for the Markov decision process defined over hyper-states as “compiling” an optimal learning strategy, which can then be “loaded” into an agent. With actions dictated by the compiled policy, the agent then behaves in a manner that is optimal with respect to its prior, which describes the distribution of environmental scenarios the agent is likely to encounter (see Section 1.4).

Chapter 1 begins by presenting several simple motivating examples of problems of optimal learning, from simple bandit problems to Markov decision processes with unknown transition probabilities. I summarize the Bayesian framework for explicitly modeling the role of uncertainty and information, and Bellman’s dynamic programming approach for computing optimal learning strategies. I provide an informal survey of reinforcement learning theory, and a relatively rigorous mathematical formulation of the problem of optimal learning for Markov decision processes with uncertain transition probabilities modeled in a Bayesian way—processes that I term *Bayes-adaptive Markov decision processes* (BAMDP’s). Chapter 1 concludes with a detailed consideration of the limitations of a conventional dynamic programming approach for computing optimal learning policies for BAMDP’s.

Chapter 2 surveys a number of approaches for addressing the explore/exploit tradeoff that have been proposed by researchers in the fields of adaptive control, decision theory, and reinforcement learning. Bellman’s Bayesian formulation of sequential decision problems (Bellman, 1956) introduced the extremely valuable concepts of information pattern, the condensed representation of information pattern by conjugate families of distributions, and the dynamic programming approach for computing optimal decision policies. Fel’dbaum examined problems of “dual control” (Fel’dbaum, 1965) for dynamical systems, following an approach similar to

Bellman's, and his perspective has been refined and extended by other control researchers (Jacobs & Patchell, 1972; Tse & Bar-Shalom, 1973). Decision theorists have pursued Bellman's approach in the context of Bayesian Markov chains (Cozzolino *et al.*, 1965; Martin, 1967; Silver, 1963), developing a rigorous distributional theory for multi-step transition probabilities and steady-state distributions, though computing these quantities, or computing optimal policies for decision processes, remains an intractable proposition for even moderately-sized problems (*i.e.*, problems with small numbers of physical states and short planning horizons). Reinforcement learning action-selection procedures have, for the most part, been driven by a desire to ensure asymptotic convergence to optimal polices, rather than by the desire to optimize cumulative reward over the duration of the learning process. In this chapter, I survey a number of action-selection heuristics, some more principled (grounded in theory) than others. In contrast to many heuristics proposed for intelligent exploration, the work described in this dissertation maintains the *full* Bayesian mathematical model throughout, it is "far-sighted" (hyperopic) in the sense that it seeks to compute policies that are optimal with respect to the *exact* and *complete* formulation of the problem.

Chapter 3 considers multi-armed bandit problems, a subclass of BAMDP's whose optimal policies may be defined in terms of Gittins indices. I present a reinforcement-learning-based algorithm for computing these indices that makes use of a particular characterization of indices as solutions to a collection of low-dimensional stopping problems. The algorithm is model-free (that is, the algorithm does not require an explicit model for process dynamics), and an example of stochastic scheduling with unknown service time distributions provides a meaningful instance of a problem in which the model-free aspect of the algorithm plays a necessary role. Chapter 3 concludes by suggesting how bandit process models and Gittins indices might be applied to general BAMDP's. By viewing BAMDP's on a time-scale of recurrence time, I derive local semi-Markov bandit process models, whose Gittins indices acknowledge some aspect of information gain.

Chapter 4 presents two, relatively direct, algorithms for approximating optimal policies for BAMDP's. Both algorithms appeal directly to the full-blown BAMDP version of Bellman's equation defined over hyperstates, but use parameterized function-approximators to preserve computational tractability. The first algorithm utilizes gradient ascent in the space of controller parameters to improve the policy and its value. I begin by developing a policy-iteration (gradient-projection) algorithm for improving stochastic policies, then translate the algorithm to an online, sample-based (model-free) scheme. I next adopt parameterized function approximators for representing policies and value functions, integrate these approximators into the stochastic policy iteration procedure, and apply the resulting "policy-gradient" algorithm to BAMDP's. The second algorithm is a relatively straightforward application of SARSA($\lambda$), a variant of reinforcement learning theory's Q-learning algorithm, to BAMDP's. Chapter 4 concludes with an example, in which I apply the policy-gradient algorithm to a problem of "optimal probing," in which the task is to identify, in an efficient way, unknown transition probabilities of a Markov decision process using online experience.

Chapter 5 considers how, at a suitable level of abstraction, BAMDP's can be cast as classical partially observable Markov decision processes (POMDP's). I show how piecewise-linearity of the value function for POMDP's generalizes to a similar property for BAMDP's, and I consider BAMDP's governed by finite-state controllers, motivated by the BAMDP/POMDP correspondence and the fact that, for "finitely-transient" POMDP's, optimal policies may be represented by finite-state automata. Chapter 5 concludes by adopting BAMDP policies expressed as finite-state *stochastic* automata, and I propose policy-improvement algorithms (similar in spirit to the policy-gradient algorithm presented in Chapter 4) that utilize Monte-Carlo techniques for gradient estimation and ascent.

Chapter 6 presents preliminary efforts to derive analytic approximations for the value of a BAMDP. If one were to consider a BAMDP with a known reward structure and some fixed policy over a finite horizon, then the value of the process is simply the inner-product between the expected number of STATE→ACTION→STATE transitions of each kind with their associated expected immediate rewards. Questions about value thus reduce to questions about the distribution of transition-frequency counts. In this chapter, I address the problem of *analytically* approximating the mean transition frequency counts associated with a Markov chain with unknown transition probabilities. I suggest a novel approach in which I model information-state components by diffusion processes, and acknowledge interdependencies between these components by defining a system of flux constraints that joint information-state trajectories must satisfy. Together, the diffusion models and flux-constraint system provide an analytical estimate for mean state-transition frequencies, and hence an approximate method for performing policy evaluation in the context of Bayesian MDP's.

Chapter 7 offers concluding comments regarding refinements and extensions of our approach. I include brief discussions of variance-reduction, generalized priors, Bayesian modeling for uncertain rewards, and optimal learning's relationship to "active learning."

In general terms, this dissertation's perspective on the problem of optimal learning seeks to approximate solutions to a problem that researchers typically dismiss as being intractable. Rather than simplifying the problem to a point where finding an exact optimal solution to the approximate problem becomes a tractable proposition, I choose to preserve the original problem, in all its complexity, and attempt to compute approximate solutions.

This dissertation makes a number of algorithmic and analytic contributions directed toward solving the problem of optimal learning:

- I develop an efficient algorithm for enumerating distinct reachable BAMDP hyperstates. This is a practical element for reducing the space requirements of conventional dynamic programming approaches that compute exact solutions (optimal learning policies).

- I develop a model-free algorithm for estimating Gittins indices for multi-armed bandit problems. The algorithm utilizes online experience in estimating the indices, which define optimal learning policies. Bandit problems are an important subclass of problems of optimal learning and have many applications (I consider an application to stochastic scheduling with unknown service-time distributions). Conversely, I show how general optimal learning problems can be modeled, in a "local" sense, by semi-Markov bandit processes, and how actions may be defined by Gittins indices for these processes.

- I develop a policy-gradient, "actor-critic" algorithm for approximating optimal policies for BAMDP's. This algorithm acknowledges the long-term aspects of information gain but has space complexity that scales polynomially with the number of physical states (space complexity does not explicitly depend upon the number of hyperstates or upon the planning horizon). I apply the algorithm to a problem of identifying an unknown Markov decision process. In the course of deriving this algorithm, I also develop a stochastic policy iteration algorithm for computing optimal policies for Markov decision processes that is interesting in its own right.

- I explore the analytical correspondence between BAMDP's and partially observable Markov decision processes. These considerations lead to the consideration of policies instantiated as finite-state stochastic automata, and I develop a Monte-Carlo gradient-based algorithm for policy improvement.

- I carry out a preliminary investigation aimed at analytically gauging the value of a BAMDP governed by a fixed policy. My approach models the evolution of information state components of the hyperstate as diffusion processes. The result is an analytic procedure for conducting approximate policy evaluation for BAMDP's, and this procedure can serve as the basis for policy improvement schemes.

Finally, I should reiterate that the problem of optimal learning is to construct policies that maximize the expected total reward that an agent derives over the entire duration of the agent's interaction with an uncertain world. By pursuing an approach that is grounded in theory, and that appeals to a complete world model that incorporates a Bayesian framework for characterizing evolving uncertainty, I develop algorithms that produce policies that exhibit performance gains over simple heuristics. Moreover, in contrast to many heuristics, the justification or legitimacy of the policies follows directly from the fact that they are clearly motivated by a complete characterization of the underlying decision problem to be solved.

# CHAPTER 1

# INTRODUCTION

## 1.1 An overview of the problem of optimal learning and its Bayesian formulation

Suppose that a medical researcher is presented with 20 critically ill patients and two new wonder drugs as yet untested (Figure 1.1). How should these drugs be tested on the patients so as to maximize the expected number of patients who are cured? The decision maker is certainly interested in determining which drug is more effective; however, his principle concern is with optimizing performance en route to this determination—his goal is to maximize the number of survivors. The optimal strategy intersperses "exploitation" steps, in which the decision maker assigns a patient to the drug with the highest estimated probability of success, with "exploration" steps in which, on the basis of observed outcomes, a drug that would be deemed inferior is administered anyway to resolve further its true potential for being the superior treatment. (This means that patients assigned treatments later in the allocation process are likely to be treated better, since they benefit from the responses of earlier patients.)

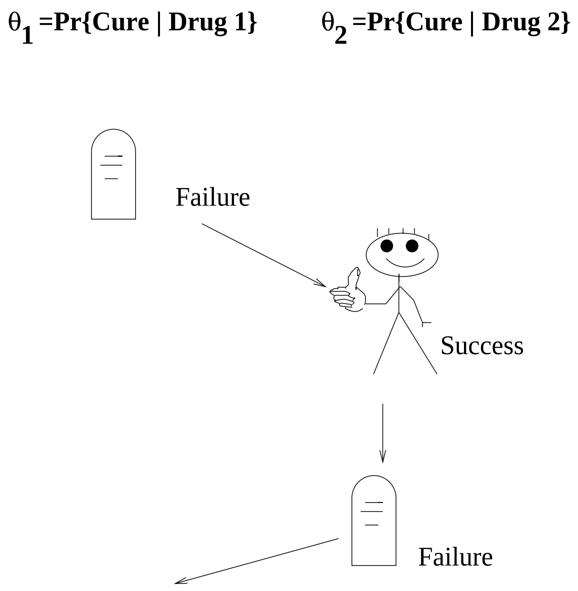


Figure 1.1: How should patients be sequentially assigned to one of two experimental treatments so as to maximize the expected number of patients who are cured?

The issue of optimally balancing exploration with exploitation is a key feature not only of this simple example[1], a two-armed bandit problem, but of general adaptive control processes as well. In a more general context, an agent might be well-advised to pursue what may be perceived as a suboptimal policy if that policy takes the agent into a previously unexplored region of state space.

For example, consider the Markov decision process (MDP) depicted in Figure 1.2. This is a two-state/two-action process, transition probabilities label arcs, and quantities within circles denote expected immediate rewards for entering particular states. The goal is to assign actions to states so as to maximize, say, the expected infinite horizon discounted sum of rewards, $E\left(\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right)$ (the "value function"), over all states.

Consider the circumstance in which the transition probabilities are not known. Given that the process is in some state, one action may seem optimal with respect to currently-perceived point-estimates of unknown parameters, while another action

[1]For a more sophisticated consideration of decision-theoretic planning to clinical management see, for example, the paper by Peek (1999).

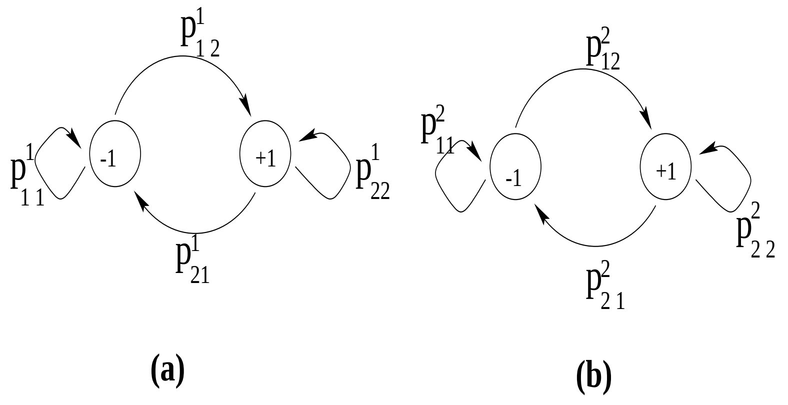


Figure 1.2: An MDP with uncertain transition probabilities: dynamics under (a) action 1 and (b) action 2 ($\pm 1$ denotes the rewards for entering right- and left-hand states).

may result in greater gain of information about these parameters and result in greater long-term reward. "Optimal learning," or "dual control," is concerned with striking a balance between these two criteria.

### 1.1.1 Bellman's formulation

If one is willing to adopt a Bayesian perspective, then the exploration-versus-exploitation issue has already been settled, in principle. A solution was recognized by Richard Bellman, among others, as early as 1956 (Bellman, 1956). A dynamic programming algorithm for Bayes-optimal policies begins by regarding "state" as an ordered pair, or "hyperstate," $(s, x)$, where $s$ is "physical state," the state of the Markov chain, and $x$ is the "information state," which summarizes past history as it relates to modeling the transitional dynamics of $s$.

For the drug-allocation example, one can model the two prospective wonder drugs as Bernoulli processes with unknown parameters, $\theta_1$ and $\theta_2$, which signify their respective probabilities of success. A Bayesian framework models our uncertainty in these parameters by a pair of probability distribution functions, or priors (Figure 1.3). The priors may be informed by previously-gathered empirical data (results with diseased mice injected with Drug 2, for example) and/or subjective criteria

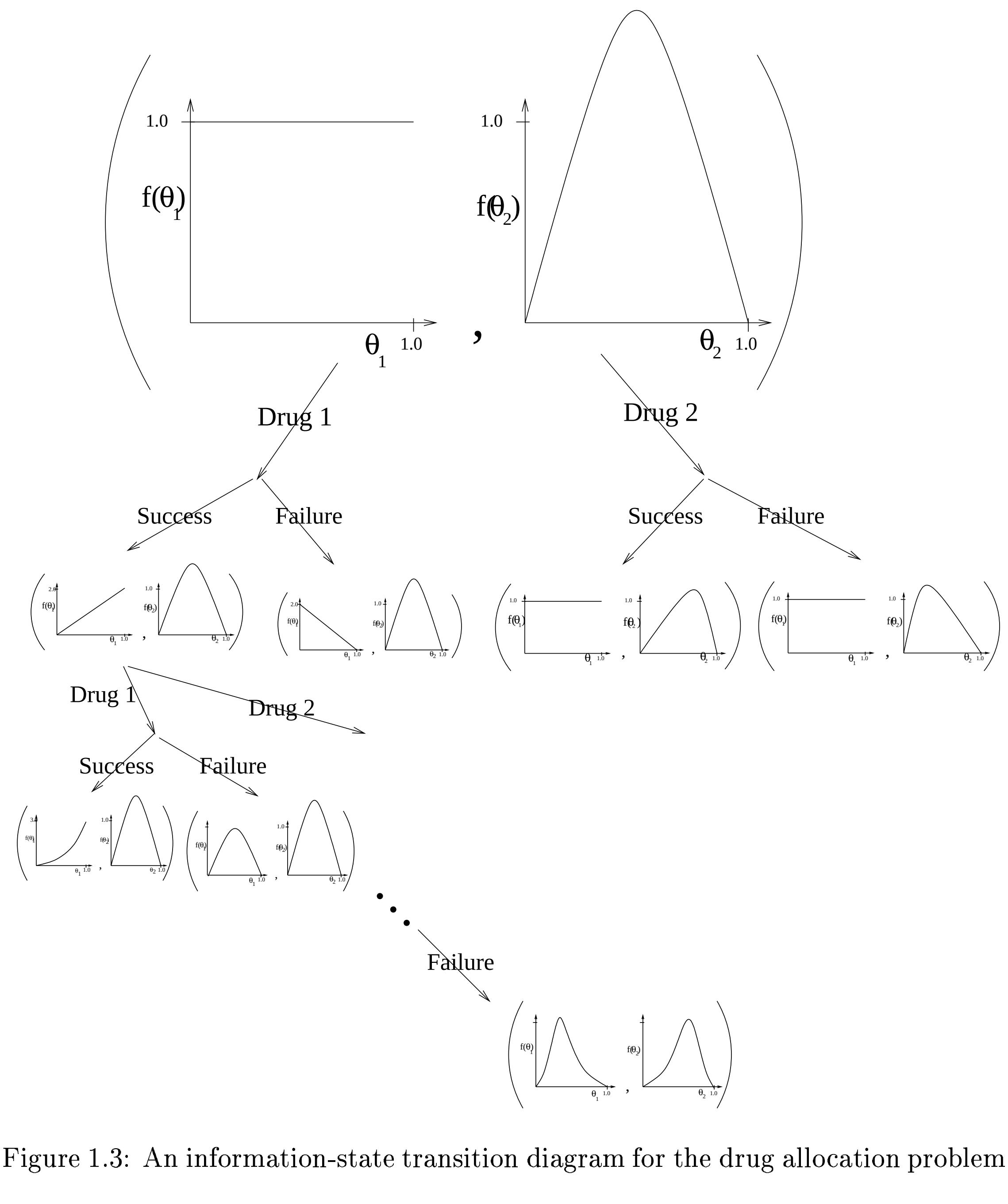


Figure 1.3: An information-state transition diagram for the drug allocation problem.

(which is more controversial). If Drug 1 were to be administered to the first patient, the expected outcome is dictated by the prior for $\theta_1$, and given an observed success or failure, we may revise the uncertainty in the efficacy of Drug 1 by applying Bayes's rule. Abstractly, we may imagine how we could exhaustively construct a transition diagram in which actions and outcomes label arcs and we identify nodes with pairs of distribution functions characterizing evolving uncertainty. One could envision terminating this tree at a depth of 20 (the number of patients), then sweeping upward from terminal leaves, "backing up" optimal values using Bellman's equation below (and noting optimal actions) along the way, until reaching the initial root node.

In general, the distributions characterizing uncertainty are continuous densities defined over continuous domains; representing these continuous densities and performing the integrations prescribed by Bayes's rule can be a computationally intensive process. In practice, it is convenient to adopt what are called "conjugate families of distributions" to model our uncertainty (DeGroot, 1970). For example, if our uncertainty in $\theta_1$ is expressed as a beta distribution parameterized by $(\alpha_1, \beta_1)$, then the posterior distribution for $\theta_1$, given an observation, is also a beta distribution, but with parameters that are incremented to reflect the observed datum (Figure 1.4 plots the beta distribution for various parameter values). The beta distribution is said to be "closed with respect to Bernoulli sampling." Using conjugate families of distributions greatly simplifies the task of representing and revising uncertainty; in the information-state transition diagram, nodes are now identified with pairs of beta-distribution *parameters*, transition probabilities between nodes follow directly from the parameter values, and Bayesian revision of the prior becomes a simple parameter increment (Figure 1.5). The process of "backing up" optimal values from the leaves of the transition tree to its root follows Bellman's optimality equation:

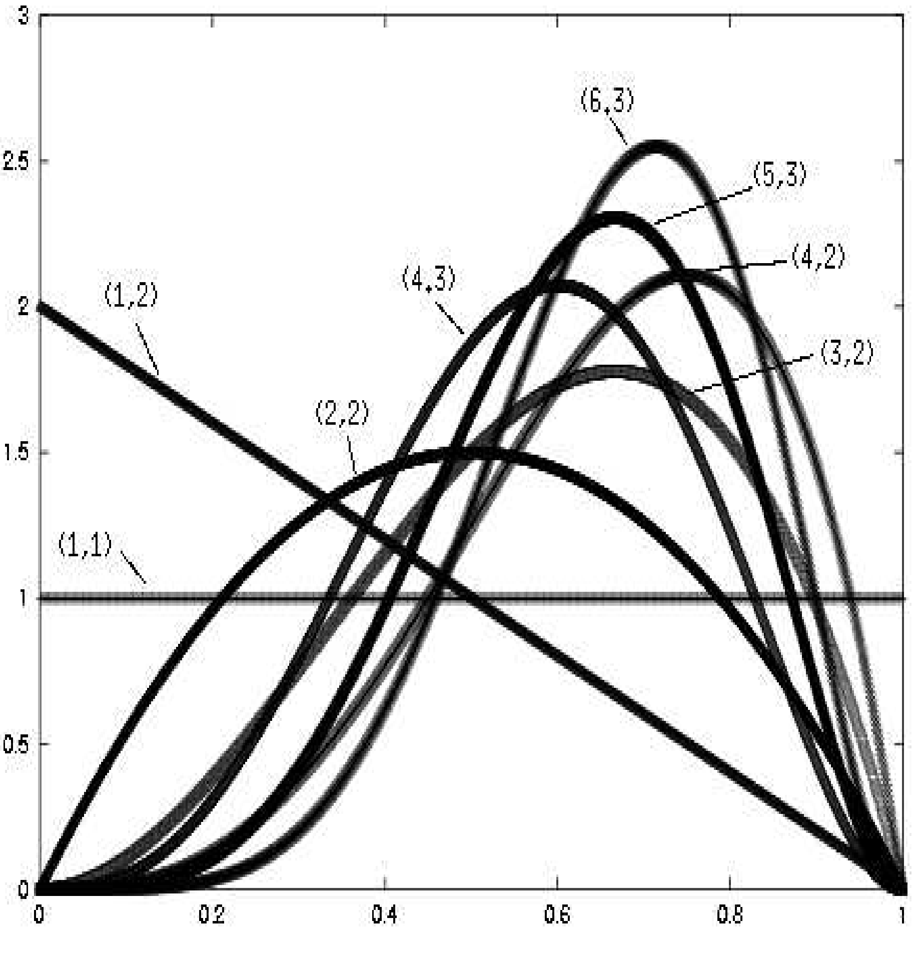


Figure 1.4: The Beta distribution density for various values of $(\alpha, \beta)$. The sequence of densities could correspond to an ("uninformed") initial uniform prior, $(\alpha, \beta) = (1, 1)$, followed by an observed failure, $(\alpha, \beta) = (1, 2)$, then a success $(2, 2)$, success $(3, 2)$, success $(4, 2)$, failure $(4, 3)$, success $(5, 3)$, and success $(\alpha, \beta) = (6, 3)$. As the number of samples increases, the posterior probability tends to be concentrated on the true state of nature.

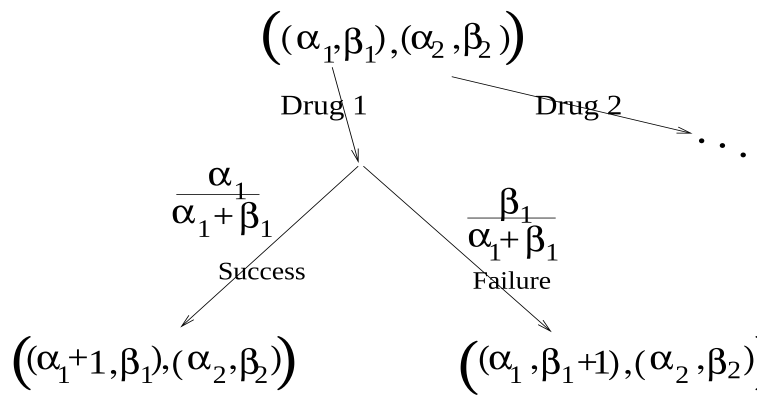


Figure 1.5: Information state transition when the prior is a member of a conjugate family of distributions.

$$\begin{aligned} V_k\left((\alpha_1,\beta_1);(\alpha_2,\beta_2)\right) &= \max\Big\{\tfrac{\alpha_1}{\alpha_1+\beta_1}\left[1+V_{k-1}\left((\alpha_1+1,\beta_1);(\alpha_2,\beta_2)\right)\right] \\ &\qquad +\tfrac{\beta_1}{\alpha_1+\beta_1}V_{k-1}\left((\alpha_1,\beta_1+1);(\alpha_2,\beta_2)\right), \\ &\qquad \tfrac{\alpha_2}{\alpha_2+\beta_2}\left[1+V_{k-1}\left((\alpha_1,\beta_1);(\alpha_2+1,\beta_2)\right)\right] \\ &\qquad +\tfrac{\beta_2}{\alpha_2+\beta_2}V_{k-1}\left((\alpha_1,\beta_1);(\alpha_2,\beta_2+1)\right)\Big\}, \end{aligned}$$

where $V_k$ denotes the optimal value with $k$ steps, or patients, remaining. (The value of terminal leaves is zero.)

We may consider a similar framework for modeling uncertainty in more general adaptive control processes that possess a physical state component to their hyperstates (the drug allocation example has no such component).

For example, consider again the MDP with unknown transition probabilities shown in Figure 1.2. If the process is in a given state and an action is taken, then the result is that the process either remains in its current physical state or jumps to the other complementary state—one observes a Bernoulli process with unknown parameter.

For the process as a whole, one observes *four* Bernoulli processes: the result of taking `action 1` in `state 1`, `action 1` in `state 2`, `action 2` in `state 1`, `action 2` in `state 2`. So if the prior probability for remaining in the current state, for each of these state-action pairs, is represented by a beta distribution (the appropriate conjugate family with regard to Bernoulli sampling), then one may perform dynamic programming in the space of hyperstates, in which the information-state component consists of four pairs of parameters specifying the beta distributions describing the uncertainty in the transition probabilities. The full hyperstate may be written as: $(s, (\alpha_1^1, \beta_1^1), (\alpha_2^1, \beta_2^1), (\alpha_1^2, \beta_1^2), (\alpha_2^2, \beta_2^2))$, where $s$ is the physical Markov chain state, and where, for example, $(\alpha_1^1, \beta_1^1)$ denotes the parameters specifying the beta distribution that represents uncertainty in the transition probability $p_{11}^1$.

In contrast to the case for bandit problems such as the problem of assigning treatments to patients, not all the Bernoulli processes are accessible to sampling at every decision stage—in order to sample the `action 1 / state 1` Bernoulli process, for example, the process must currently reside in `state 1`, hence the physical state component to the hyperstate. For bandit problems, there is no physical state component to the hyperstate; the hyperstate consists solely of information state, and under certain conditions this special property allows optimal strategies for bandit problems to be expressed concisely and computed with a reasonable amount of effort (Gittins & Jones, 1974; Katehakis & Veinott, 1987).

Figure 1.6 shows part of the hyperstate transition diagram associated with the Bayes-adaptive Markov decision process of Figure 1.2. We have written the information state parameters in matrix form to highlight their observed correspondence to transition counts modulo the prior; *i.e.*, if we subtract the information state associated with the prior from any given hyperstate's information state component, the result may be interpreted as the number of transitions of each type observed in transit from the initial hyperstate to the given hyperstate.

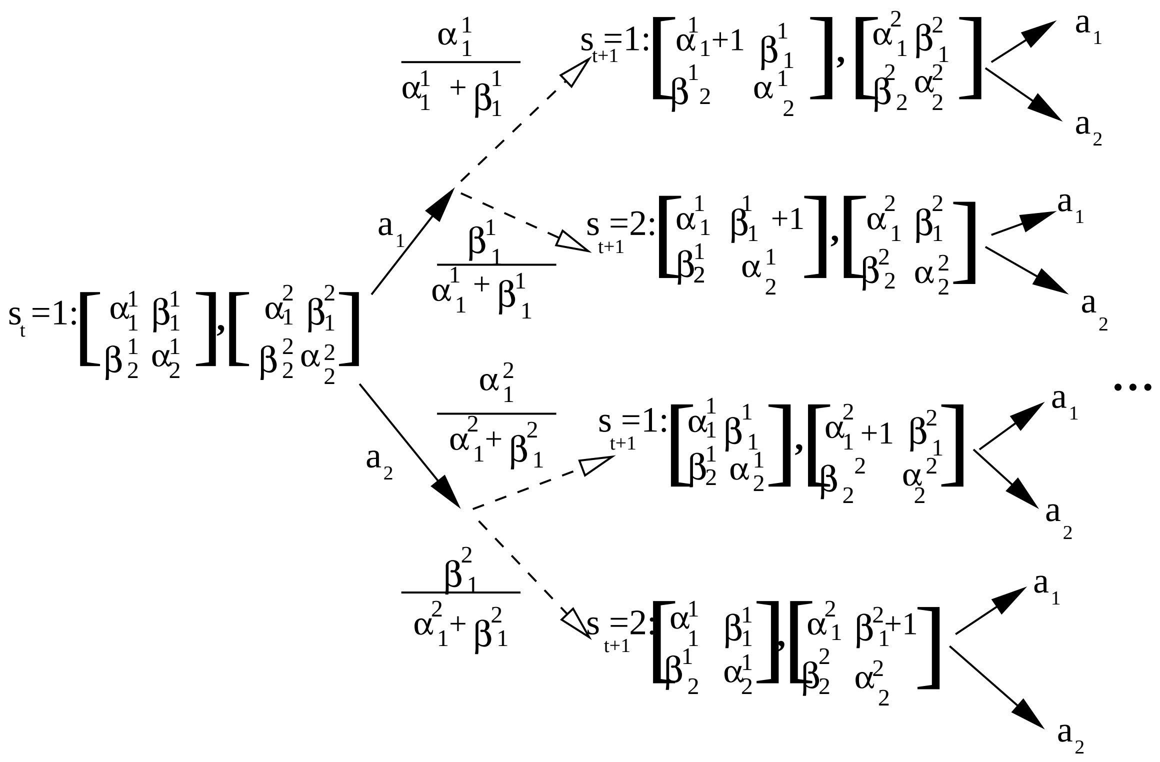


Figure 1.6: The hyperstate transition diagram associated with the Bayes-adaptive Markov decision process of Figure 1.2. The initial hyperstate may be identified with physical state 1 and an information state specifying prior uncertainty in physical-state transition probabilities. Subsequent actions and observed physical state transitions correspond to transitions in the tree (note how hyperstate parameters define the transition probabilities, which label dashed segments leading from hyperstates). The number of reachable hyperstates grows exponentially with the horizon.

An optimality equation may be written in terms of the hyperstates; for example, the $\left(1; \begin{bmatrix} \alpha_1^1 & \beta_1^1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2 & \beta_1^2 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right)$th component of the optimal value function must be consistent with local transitions to successor hyperstates and their values:

$$V\left(1; \begin{bmatrix} \alpha_1^1 & \beta_1^1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2 & \beta_1^2 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right) =$$

$$\max\left\{ \frac{\alpha_1^1}{\alpha_1^1+\beta_1^1}\left[r_{11}^1 + \gamma V\left(1; \begin{bmatrix} \alpha_1^1+1 & \beta_1^1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2 & \beta_1^2 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right)\right]\right.$$

$$+\frac{\beta_1^1}{\alpha_1^1+\beta_1^1}\left[r_{12}^1 + \gamma V\left(2; \begin{bmatrix} \alpha_1^1 & \beta_1^1+1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2 & \beta_1^2 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right)\right],$$

$$\frac{\alpha_1^2}{\alpha_1^2+\beta_1^2}\left[r_{11}^2 + \gamma V\left(1; \begin{bmatrix} \alpha_1^1 & \beta_1^1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2+1 & \beta_1^2 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right)\right]$$

$$\left.+\frac{\beta_1^2}{\alpha_1^2+\beta_1^2}\left[r_{12}^2 + \gamma V\left(2; \begin{bmatrix} \alpha_1^1 & \beta_1^1 \\ \beta_2^1 & \alpha_2^1 \end{bmatrix}, \begin{bmatrix} \alpha_1^2 & \beta_1^2+1 \\ \beta_2^2 & \alpha_2^2 \end{bmatrix}\right)\right]\right\}.$$

For MDP's with more than two physical states, sampling becomes multinomial, for which the appropriate conjugate family of distributions is Dirichlet, and the current formulation generalizes in a straightforward way. We should also mention that the concept of a conjugate family of distributions can be generalized. One can consider mixtures of conjugate distributions, or "extended natural conjugate distributions" (Martin, 1967), which provide more degrees of freedom for expressing our prior uncertainty and allow for nonzero correlation between rows of the generalized transition matrix.[2] However, the central issue that must be confronted is what Bell-

[2]A Dirichlet prior assumes independence between rows of the generalized transition matrix. For a matrix with $N$ columns, one would like to select Dirichlet parameters associated with each row to match the prior estimates of the means and variances of the $N$ individual $p_{ij}^a$'s. However, this would require $2N-1$ parameters, and the Dirichlet prior supplies us with only $N$. A practical approach for assigning a prior from empirical data proceeds by fitting the means exactly, then by performing a least-squares fit of the remaining Dirichlet parameter to match the variances of the $p_{ij}^a$'s (Silver, 1963).

man calls the "menace of the expanding grid"—the number of reachable hyperstates (in Figure 1.6, for example) grows exponentially with the time horizon.

### 1.1.2 Limitations of certainty-equivalence

How does one proceed if one is constrained to practical amounts of computation and is willing to settle for an approximate solution? We could truncate the hyperstate transition diagram at some shorter and more manageable horizon, compute approximate terminal values by replacing the distributions with their means, and proceed with a receding-horizon approach: Starting from the approximate terminal values at the horizon, perform a backward sweep of dynamic programming, computing an optimal policy. Take the initial action of the policy, then shift the entire computational window forward one level and repeat. One can imagine a limiting, degenerate version of this receding horizon approach in which the horizon is zero; that is, use the means of the current distributions to calculate an optimal policy, take an "optimal action," observe a transition, perform a Bayesian modification of the prior, and repeat. This heuristic was suggested by Cozzolino *et al.* (1965), and has appeared more recently (Dayan & Sejnowski, 1996). However, as was noted in Cozzolino *et al.*, "...the trade-off between immediate gain and information does not exist in this heuristic. There is no mechanism that explicitly forces unexplored policies to be observed in early stages. Therefore, if it should happen that there is some very good policy which a priori seemed quite bad, it is entirely possible that this heuristic will never provide the information needed to recognize the policy as being better than originally thought..."

This comment refers to what is regarded in engineering literature as a problem of "identifiability" associated with certainty-equivalence controllers in which a closed-loop system evolves identically for both true and false values of the unknown parameters; that is, certainty-equivalence control may make some of the unknown

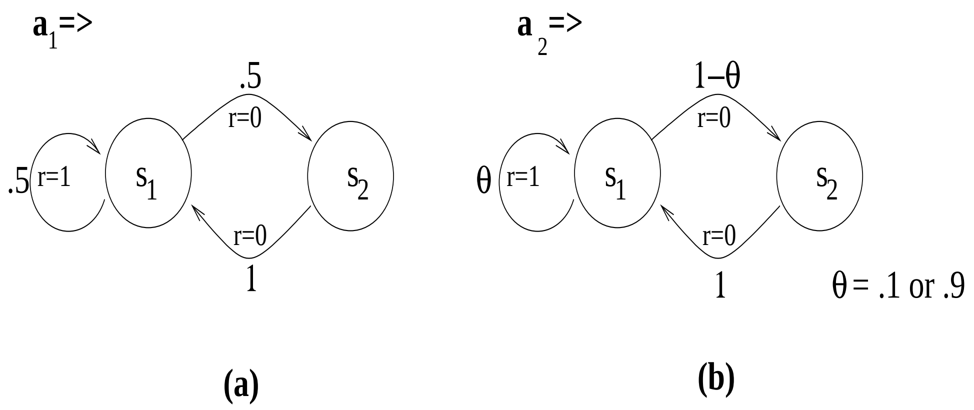


Figure 1.7: Transitions/rewards under (a) action $a_1$ and (b) action $a_2$. For values of the prior parameter $\xi = Pr\{\theta = .9\} < \frac{1}{2}$, a certainty-equivalence approach will repeatedly choose action $a_1$, which will be the worst policy with probability $\xi$.

parameters invisible to the identification process and lead one to choose the wrong action repeatedly.

**Example (Bokar & Varaiya, 1979).** Consider the MDP depicted in Figure 1.7, in which uncertainty exists for the transition probabilities for transitions from state $s_1$ in response to executing action $a_2$; either $\theta = p^{a_2}_{s_1 s_1} = .1$ or $\theta = .9$. Rewards are zero but for transitions from $s_1$ back to itself, which have REWARD=1.

Suppose the prior for $\theta$ is:

$$\begin{aligned} Pr\{\theta = .9\} &= \xi \\ Pr\{\theta = .1\} &= 1 - \xi. \end{aligned}$$

The certainty-equivalent estimate for $p^{a_2}_{s_1 s_1}$is:

$$\begin{aligned} \overline{p}^{a_2}_{s_1 s_1} &= \xi\,(.9) + (1-\xi)\,.1 \\ &= .8\xi + .1, \end{aligned}$$

which implies that for $\xi < \frac{1}{2}$, $\overline{p}^{a_2}_{s_1 s_1} < \frac{1}{2}$.

Thus if the prior probability for $p^{a_2}_{s_1 s_1}$being .9 is less than $\frac{1}{2}$, then the certainty-equivalence view is that action $a_1$ is the best action. Executing $a_1$ will never change the posterior distribution for $\theta$, and $a_1$ will be executed forever. If the prior is an accurate model of uncertainty, then with likelihood $\xi$ the true state of nature will be $\theta = .9$, in which case the best action is $a_2$. So with a probability that may be arbitrarily close to a half, a certainty-equivalence approach leads to a policy that does precisely the wrong thing.

## 1.2 Reinforcement learning: An informal survey

Reinforcement learning algorithms, such as the method of temporal differences (TD) (Sutton, 1988) and Q-learning (Watkins, 1989), have been advanced as models of animal learning, originally motivated and inspired by the behavioral paradigms of classical and instrumental conditioning. These algorithms have subsequently proved useful in solving certain problems of prediction and control encountered by general adaptive real-time systems or agents embedded in stochastic environments. Supporting theory and applications have reached a stage of development that is relatively mature (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). Connections have been established with stochastic dynamic programming and heuristic search, and a mathematical framework, grounded in the classical theory of stochastic approximation, has led to new and improved proofs of convergence. Researchers have customarily focused their attention upon asymptotic learning of maximally-efficient strategies, and not on the "optimal learning" of these strategies—optimal learning explicitly acknowledges and addresses how well one performs en route to the determination of the (asymptotically-learned) optimal policy.

Readers unfamiliar with reinforcement learning theory may wish to consult other sources, including the survey paper by Kaelbling, Littman, and Moore (1996), the immanently readable introductory text by Sutton and Barto (1998), and the more mathematically-inclined book by Bertsekas and Tsitstiklis (1996).

### 1.2.1 Markov decision processes

Consider a system whose dynamics are described by a finite state Markov chain with transition matrix $P$, and suppose that at each time step, in addition to making a transition from state $s_t = i$ to $s_{t+1} = j$ with probability $P_{ij}$, the system produces a randomly determined reward, $r_{t+1}$, whose expected value is $R(i)$. The *evaluation function*, $V$, maps states to their expected, infinite-horizon discounted returns,

$$V(i) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = i\right\},$$

and $V(\cdot)$ may also be shown to be the unique solution to a system of linear equations describing local consistency: for $i = 1$ to $N$,

$$V(i) = R(i) + \gamma \sum_j P_{ij} V(j).$$

In addition, for each state $i$, suppose that there is a set of feasible actions, $\mathcal{A}_i$, from which to choose, and that choosing an action, $a$, determines the transition probabilities and reward distribution associated with that state. The resulting Markov chain is called a *Markov decision process* (MDP). A (stationary) *policy* is a mapping of states to actions, and one may think of $P$ above as the transition matrix associated with some particular policy. An *optimal* policy is one that optimizes the value function over all states; the associated *optimal* value function is the unique solution to Bellman's optimality equation: for $i = 1$ to $N$,

$$V(i) = \max_{a \in \mathcal{A}_i} \left[ R(i,a) + \gamma \sum_{j=1}^{N} P_{ij}(a) V(j) \right]. \qquad (1.1)$$

An optimal policy is determined by the optimal value function through Equation 1.1; that is, if $V$ is the optimal value function, then for each state $i$, an optimal action for that state is the action that achieves equality in Equation 1.1.

Standard methods for solving Bellman's equation include value iteration, policy iteration, and linear programming—see, for example, Ross (1983) or Bertsekas (1987).

### 1.2.2 TD(0)

Recall from the previous section the system of linear equations that determines a set of values for a Markov chain with rewards: for $i = 1$ to $N$,

$$V(i) = R(i) + \gamma \sum_j P_{ij} V(j). \tag{1.2}$$

Since the right-hand side, viewed as an operator acting on $V(\cdot)$, is a contraction, successive approximation is one viable computational scheme for finding the solution: for $i = 1$ to $N$,

$$V^{(k+1)}(i) = R(i) + \gamma \sum_j P_{ij} V^{(k)}(j).$$

TD(0) (Sutton, 1988) is a stochastic approximation method for finding solutions to Equation 1.2; it proceeds by taking very small steps in the directions suggested by the "instantaneously-" sampled version of successive approximation; *i.e.,* having observed a transition for state $i$ to $j$ with reward $r$, one's instantaneous view of the right-hand side of the successive approximation recursion is $r + \gamma V^{(k)}(j)$. TD(0) takes a small step in the direction $r + \gamma V^{(k)}(j) - V(i)$:

$$\begin{aligned} V^{(k+1)}(i) &= V^{(k)}(i) + \alpha_k [r + \gamma V^{(k)}(j) - V^{(k)}(i)] \\ &= (1 - \alpha_k) V^{(k)}(i) + \alpha_k [r + \gamma V^{(k)}(j)]. \end{aligned}$$

The update is severely "under-relaxed" ($\alpha_k$ is very small) so as to, over the long run, average out the sampling noise.[3] The fact that the expected value of the instantaneous sample is equal to successive approximation's right-hand side implies, by the theory of stochastic approximation, that if step-sizes $\alpha_k$ diminish at the appropri-

[3]Standard stochastic approximation conditions on step size to assure convergence (with probability 1) are that $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

ate rate and other sensible assumptions hold, then the sequence of value-function estimates generated by the TD(0) procedure will converge to the true solution with probability 1 (Jaakkola, Jordan, & Singh, 1994).

### 1.2.3 TD($\lambda$)

One way (Watkins, 1989) of viewing the TD($\lambda$) algorithm (Sutton, 1988) is that it updates its current value function estimate, $\widetilde{V}(s_t)$, in the direction of a weighted combination of the following (infinite) family of estimators:

$$X^{(k)} \overset{def}{=} r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k \widetilde{V}(s_{t+k}) \quad k = 1, 2, \ldots$$

For large values of $k$ (such that $\gamma^k$ is small), the "truncated corrected return," $X^{(k)}$, is approximately equal to the sampled discounted return, which is an unbiased estimate for $V(s_t)$, and has variance equal to the variance of accumulated discounted rewards encountered along sample paths. For smaller values of $k$, the portion of total variance due to the variance of summed, discounted rewards beyond time horizon $k$ is replaced by the variance of $\widetilde{V}(s_{t+k})$. If the approximate value function is equal to the true value function, then the variance of this "tail" will be less than the variance of the tail of sampled returns; if the approximate value function is in error, then in general $X^{(k)}$ will be a biased estimator.

In TD($\lambda$), the estimators, $X^{(k)}$, are combined via

$$X_w = \sum_{k=1}^{\infty} w_k X^{(k)},$$

where $w_k = \lambda^{k-1}(1-\lambda)$, and $\widetilde{V}(s_t)$ is updated via

$$\widetilde{V}^{new}(s_t) = (1-\alpha)\,\widetilde{V}^{old}(s_t) + \alpha X_w.$$

The weighting function, $w_k$, is such that, for small values of $\lambda$, the resulting estimate $X_w$ is more heavily weighted toward $X^{(k)\prime}s$ with small values of $k$, estimators with potentially small variance but high bias.

Watkins has suggested that a reasonable strategy might be to apply TD($\lambda$) with $\lambda$ near 1 at the outset of learning, so as to avoid the ill-effects of using biased $\widetilde{V}$'s, then as confidence in the $\widetilde{V}$'s grows, to slowly reduce $\lambda$ to reduce variance. The decaying exponential form of the weighting function gives rise to an update rule that can be implemented incrementally, but there is no reason to presume that the "best" combination of estimators would be weighted in this way for *any* value of $\lambda$, in fact one might well suspect that the best weighting scheme would (at least) vary with state and time.

Our presentation of TD($\lambda$) has adopted a theoretical "forward view" which decides how to update each state by looking forward to future rewards and states. In practice, TD($\lambda$) is implemented by maintaining an "eligibility trace" for each state, an exponentially-decaying count of the number of state visitations. At each step, traces for all states decay by a scale-factor $\gamma\lambda$, and the trace for the state visited at the given step is incremented by 1; that is, upon observing a transition, $s_t \rightarrow s_{t+1}$, the eligibility traces for all states are updated via:[4]

$$e_{t+1}(s) = \begin{cases} \gamma\lambda e_t(s) + 1 & \text{if } s = s_{t+1} \\ \gamma\lambda e_t(s) & \text{otherwise.} \end{cases}$$

[4]This eligibility trace update implements "accumulating" traces. Alternatively, a state's eligibility may be reset to 1 when that state is entered. This "replacing" trace is implemented by

$$e_{t+1}(s) = \begin{cases} 1 & \text{if } s = s_{t+1} \\ \gamma\lambda e_t(s) & \text{otherwise.} \end{cases}$$

Experimental evidence suggests that replacing traces can produce superior learning rates (Singh & Sutton, 1996).

Supposing that the observed state transition also generates immediate reward $r$, the TD error for state value prediction,

$$\delta = r + \gamma \tilde{V}(s_{t+1}) - \tilde{V}(s_t),$$

gets "smeared" back to all recently visited states via

$$\tilde{V}^{new}(s) = \tilde{V}^{old}(s) + \alpha\delta e(s) \qquad \forall s.$$

Note that a naive implementation of TD($\lambda$) requires updating every state's value estimate and eligibility trace after each transition—practical implementations may update only recently-visited states. For versions of TD($\lambda$) that incorporate parameterized function approximators (see Section 1.2.6), these computational issues essentially disappear.

### 1.2.4 Q-learning

Recall from Section 1.2.1 the system of *nonlinear* equations that determines a set of *optimal* values for a Markov decision process; *i.e.,* Bellman's equation: for $i = 1$ to $N$,

$$V(i) = \max_{a \in \mathcal{A}_\rangle} \left[ R(i,a) + \gamma \sum_{j=1}^{N} P_{ij}(a) V(j) \right]$$

One can show that the right-hand side of this equation, viewed as an operator, is also a contraction, and so a successive approximation method will converge to the solution. Blindly attempting to apply a stochastic approximation approach using this successive approximation recursion, however, is complicated by the presence of the "max" operation. What is the meaning of "instantaneous sample of $\max_a \left\{ R(i,a) + \gamma \sum_j P_{ij}(a) V^{(k)}(j) \right\}$?" The issue can be sidestepped by shifting the focus from the estimation of optimal state values to the estimation of optimal state/action values, which are termed optimal "Q-values" (Watkins, 1989).

The Q-function of a state/action pair, with regard to a value-function estimate, is the expected infinite-horizon return when, starting in the given state, the given action is applied, and actions thereafter are prescribed by a policy having the given value-function; that is, $Q_V(i,a) = R(i,a) + \gamma \sum_j P_{ij}(a)V(j)$, the quantity appearing within the scope of the "max" operator in Bellman's equation. Informally, the Q-function with regard to a policy is the value of the policy if the policy's initial action is perturbed.

In terms of Q-functions, Bellman's equation is just $\max_{a'} Q^*(j,a') = V^*(j)$, where the star signifies *optimal* value functions. Therefore, by the definition of $Q$,

$$Q^*(i,a) = R(i,a) + \gamma \sum_j P_{ij}(a) \max_{a'} Q^*(j,a'),$$

which suggests the successive approximation recursion:

$$Q^{(k+1)}(i,a) = R(i,a) + \gamma \sum_j P_{ij}(a) \max_{a'} Q^{(k)}(j,a').$$

The meaning of "instantaneous sample of the right-hand side" for this equation is less problematic than that for the original form of Bellman's equation. The introduction of $Q$ allows one, in effect, to interchange the order of expectation and maximization. Observing a transition from state $i$ to $j$ and reward $r$ upon executing action $a$ admits an instantaneous view of the right-hand side as $r + \gamma \max_{a'} Q^{(k)}(j,a')$, which suggests a stochastic approximation recursion, or "backup," of the form

$$\begin{aligned} Q^{(k+1)}(i,a) &= Q^{(k)}(i,a) + \alpha_k \left[ r + \gamma \max_{a'} Q^{(k)}(j,a') - Q^{(k)}(i,a) \right] \\ &= (1-\alpha_k) Q^{(k)}(i,a) + \alpha_k \left[ r + \gamma \max_{a'} Q^{(k)}(j,a') \right]. \end{aligned} \tag{1.3}$$

As with TD methods, the step-size, $\alpha_k$, should satisfy stochastic approximation conditions to assure convergence. Various schemes have been proposed for parameterizing the learning-rate to accelerate the convergence process; for example, the "search then converge" method of Darken and Moody (1992) maintains a set of counters, $n_t(i,a)$, which count the number of backups of each $Q(i,a)$ up to time $t$. The learning rate is then defined by

$$\alpha_t(i,a) = \frac{\alpha_0 \tau}{\tau + n_t(i,a)},$$

where $\alpha_0$ is some initial learning rate (*e.g.*, 0.5), and $\tau$ is a parameter (a typical value is 300).

Theory (Bertsekas & Tsitsiklis, 1996) implies that, if each action is tried in each state an infinite number of times, then $Q^{(k)}$ converges to $Q^*$. In practice, the evolving current estimate of $Q^*$ is often used to guide the selection of actions, resulting in a gradual focusing of backup operations to states encountered along sample paths generated through the application of actions comprising optimal policies (Barto, Bradtke, and Singh, 1991, includes a discussion of the ensuing computational efficiency). The goal of action-selection is to strike a balance between refining estimates for actions currently thought to be optimal for states that lie on optimal trajectories and exploring regions of the state-action space for which there is a high degree of value-function uncertainty.

A Q-learning computational cycle proceeds as follows (assuming that the system is currently in state $i$):

- Choose an action, $a$, via some designated action-selection procedure (see Section 2.3).
- Observe the state transition, $i \rightarrow j$, under action $a$, generating immediate reward $r$.
- Perform the backup operation (Equation 1.3) to update the Q-value associated with state $i$ / action $a$.

### 1.2.5 SARSA

SARSA replaces Q-learning's backup operation (Equation 1.3) with

$$Q^{(k+1)}(i,a) = (1-\alpha_k)\, Q^{(k)}(i,a) + \alpha_k \left[ r + \gamma Q^{(k)}(j,a') \right],$$

where $(i,a) \xrightarrow{r} (j,a')$ is an observed transition from one state-action pair to the next. SARSA's backup operation is based upon data generated by the actions actually executed, it is an "on-policy" method. In contrast, Q-learning, an "off-policy" method, updates its value in a direction determined by $\max_{a'} Q(j,a')$, though action-selection may direct the agent to execute some action at state $j$ other than the maximizing $a'$. The convergence of SARSA with probability 1 to an optimal policy (given adequate sampling of state-action values) has been proved by Singh, Jaakkola, Littman, and Szepesvari (2000).

### 1.2.6 Parameterized function approximation

The preceeding reinforcement learning algorithms maintain value estimates in tabular form, and each observed state-transition gives rise to an update of a single entry in the table. For problems with a large number of states, this approach may not be feasible—storage requirements may be prohibitive. Aside from this issue, we would like to accelerate learning by generalizing value-updates to states or state-action pairs that are not actually sampled but share common features to those that are. One way of doing this is to combine the basic reinforcement learning procedures with existing numerical function approximation procedures for performing extrapolation and interpolation. Figure 1.8 depicts the basic process of parameterized function-approximation.[5] First, the state $s_t$ (or state-action pair) is projected

[5]We note that maintaining value function estimates for each state in a table may be regarded as "function approximation" in the sense that table entries approximate true values; however, the tabular representation has sufficient degrees-of-freedom to "fit" value function *estimates* perfectly—a table is an unbiased function approximator. In general, the utilization of parameterized function approximators introduces additional approximation error. The estimated value function may not fall within the span of functions representable by the assumed functional form of the parameter-

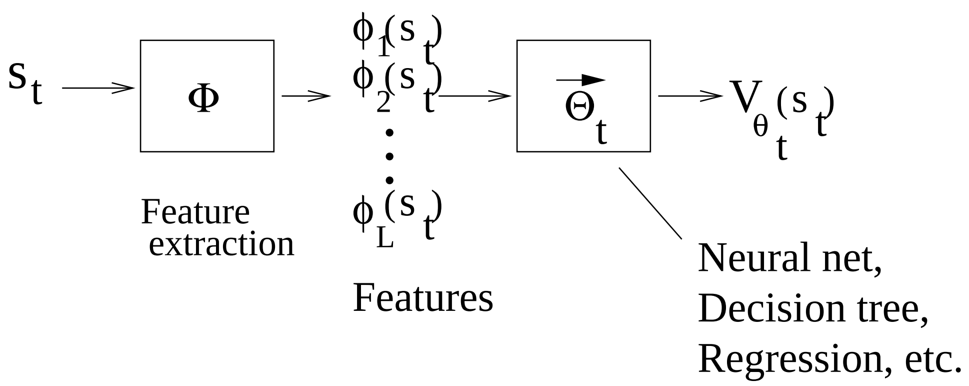


Figure 1.8: A schematic representation of the function-approximation process.

onto some low-dimensional set of "features"—in general the number of features, $L$, is much smaller than the number of states. The feature set then serves as input to some function-approximation architecture (such as a neural network or linear-approximator) that is parameterized by a vector, $\vec{\theta}_t$, whose dimension is also small relative to the number of states. The storage requirements are thus reduced from the tabular case (from being on the order of the number of states) to being on the order of the dimension of $\vec{\theta}$, and changing a single parameter value, $\theta_i$, now affects value estimates for many states—function approximation effects a generalization over states that share similar features. The selection of a particular set of features is problem-dependent; it may be informed by prior knowledge and is often constructed through a process of trial-and-error.

With a function-approximation architecture in place, we can now consider various supervised learning schemes for adjusting the function-approximator's parameter vector, $\vec{\theta}$. We may regard reinforcement learning's backup value as a "target" for a supervised-learning instance. Note that since backup values include evolving value estimates, these targets will be non-stationary. Parameter-adjustment schemes seek to minimize some error measure, for instance, the squared error, $(V^\pi(s_t) - V_\theta(s_t))^2$,

ized approximator. The adoption of parameterized approximators with reduced degrees-of-freedom introduces bias into the approximation process, giving rise to interpolative errors, which in some instances can cause divergence of value function estimates (Baird, 1995; Tsitsiklis & Van Roy, 1997).

weighted by the process's state distribution ($V^{\pi}$ denotes the value function associated with policy $\pi$). Replacing the true, unknown target value, $V^{\pi}(s_t)$, with a TD backup value as an approximate target, $r + \gamma V_{\theta}(s_{t+1})$, and computing the gradient of squared error with respect to $\overrightarrow{\theta}$ (disregarding the ersatz target's dependence upon $\theta$),

$$\nabla_{\theta} \left(V^{\pi}(s_t) - V_{\theta}(s_t)\right)^2 \cong -\left(r + \gamma V_{\theta}(s_{t+1}) - V_{\theta}(s_t)\right) \nabla_{\theta} V_{\theta}(s_t),$$

leads to a gradient-descent procedure for adjusting $\overrightarrow{\theta}$; *i.e.*, we take a small step in the direction suggested by the negative of the approximate error-gradient. For a TD($\lambda$) version of this procedure, we replace $\nabla_{\theta} V_{\theta}(s_t)$ with an exponentially-decaying "eligibility-trace" of past gradients, which can be implemented incrementally by

$$e_t = \gamma \lambda e_{t-1} + \nabla_{\theta} V_{\theta}(s_t).$$

This approach takes a particularly simple form when the function approximator is linear in its parameters. For example, consider the case in which value function approximators are linear combinations of features:

$$V_{\theta}(s) = \sum_{l=1}^{L} \theta_l \phi_l(s).$$

In this instance, the gradient, $\nabla_{\theta} V_{\theta}(s)$, is simply the feature vector $\overrightarrow{\phi}(s)$, and for TD($\lambda$), the eligibility-trace is a decaying trail of previously encountered features. It has been shown (Tsitsiklis & Van Roy, 1997) that TD($\lambda$)'s linear function approximator parameters converge, and that the parameter vector to which they converge has an associated approximate value function whose mean-squared error is within a scale factor, $\frac{1-\gamma\lambda}{1-\gamma}$, of the best mean-squared error that can be achieved.

### 1.2.7 Gradient-descent SARSA($\lambda$)

Function approximation methods can be applied to problems of control as well. In this case, our parameterized functions attempt to provide good approximations to estimated Q-values. The approach is similar to that for TD, but with state-action backups providing ersatz targets. For example, SARSA's backup is $r + \gamma Q_\theta(s_{t+1}, a_{t+1})$, which leads to the gradient-descent update equation:

$$\overrightarrow{\theta}^{new} = \overrightarrow{\theta}^{old} + \alpha \left(r + \gamma Q_\theta(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)\right) \nabla_\theta Q_\theta(s_t, a_t).$$

SARSA($\lambda$) replaces the gradient term in this update with an exponentially-decaying trace of gradients, which can be incremented incrementally via:

$$e_t = \gamma \lambda e_{t-1} + \nabla_\theta Q_\theta(s_t, a_t).$$

### 1.2.8 Reinforcement learning and optimal learning

Reinforcement learning approaches have recognized the dual-effects of control, at least in the sense that one must occasionally deviate from a greedy policy to ensure a search of sufficient breadth. But reinforcement-learning action selection procedures are not fundamentally driven by real notions of dual control or optimal learning, their exploration schemes aspire to be practical methods for avoiding the unrealistic amount of sampling and search that would be required if one were to strictly adhere to the theoretical sufficient conditions for convergence—that all state-action pairs be considered infinitely many times.

One representative action selection procedure (discussed in further detail in Section 2.3.3) suggests choosing an action via the Boltzmann distribution, which involves a parameter, "temperature," that is initialized to a relatively high value, resulting in random exploration of prospective actions. As computation proceeds, the Boltzmann scheme gradually lowers its temperature, in effect raising the probability of selecting actions with higher estimated value, but in general, the resulting sequence of actions generated en route to the optimal policy is not an optimal one in the sense of dual control—for many optimal learning problems, initial random exploration would be catastrophic, and in optimal learning, the decision-making agent is obliged to enjoy or endure the consequences of actions experienced in the process of convergence to an asymptotically-learned optimal policy.

## 1.3 Mathematical formulation of the problem of optimal learning

Summarizing Martin (1967), this section presents a brief summary of the Bayesian formulation of Markov decision processes with uncertainty in their state-transition probabilities; we shall henceforth refer to these processes as *Bayes-adaptive Markov decision processes* (BAMDP's).

### 1.3.1 Transition matrix uncertainty

Let $p_{ij}^a$ denote the probability of transition from state $i$ to state $j$ under the influence of action $a$, and define the *generalized transition matrix* as:

$$P = \begin{bmatrix} p_{11}^1 & p_{12}^1 & \cdots & p_{1N}^1 \\ p_{11}^2 & p_{12}^2 & \cdots & p_{1N}^2 \\ \vdots & \vdots & & \vdots \\ p_{11}^{A_1} & p_{12}^{A_1} & \cdots & p_{1N}^{A_1} \\ p_{21}^1 & p_{22}^1 & \cdots & p_{2N}^1 \\ p_{21}^2 & p_{22}^2 & \cdots & p_{2N}^2 \\ \vdots & \vdots & & \vdots \\ p_{21}^{A_2} & p_{22}^{A_2} & \cdots & p_{2N}^{A_2} \\ \vdots & \vdots & & \vdots \\ p_{N1}^1 & p_{N2}^1 & \cdots & p_{NN}^1 \\ p_{N1}^2 & p_{N2}^2 & \cdots & p_{NN}^2 \\ \vdots & \vdots & & \vdots \\ p_{N1}^{A_N} & p_{N2}^{A_N} & \cdots & p_{NN}^{A_N} \end{bmatrix}.$$

$A_i$ is the number of admissible alternative actions when in state $i$, and let $A = \sum_i A_i$ be the number of rows of $P$. Similarly, define a matrix, $R$, specifying a set of expected one-step immediate rewards, $r_{ij}^a$. Together, $P$ and $R$ suffice to specify a Markov decision process.

Now suppose that $P$ is a random variable with prior distribution $H(P|\psi)$. $H$ is a function of the $A(N-1)$ independent elements of $P$, and the distribution parameter, $\psi$, is in general a point in a multidimensional Euclidean space.

If we sample the process (observe a transition, "$x$"), then we can update $H$ using Bayes's rule:

$$dH(P|\psi, x) = \frac{l(x|P)dH(P|\psi)}{\int_P l(x|P)dH(P|\psi)};$$

*i.e.*, the posterior density is proportional to the likelihood function (the conditional density, $l(x|P)$) times the prior density, $dH(P|\psi)$.

### 1.3.2 Conjugate families of distributions

Let $\mathcal{H}$ be a family of distributions indexed by $\psi$. If $H(P|\psi) \in \mathcal{H}$, and $H(P|\psi, x) \in \mathcal{H}$ for every $x$ and $\psi$, then $\mathcal{H}$ is "closed" with respect to the sampling rule which determines $l(x|P)$. $\mathcal{H}$ is a *conjugate family of distributions.* We denote the mapping induced by Bayes's rule as $\psi' = T(\psi|x)$ or, for an observation of an state $i$ to state $j$ transition under action $a$,

$$\psi' = T_{ij}^a(\psi).$$

For Markov decision processes, the state transition samples one observes are governed by multinomial distributions, one for each row of $P$. It can be shown that an appropriate conjugate family of distributions in this case is the "matrix beta", or Dirichlet, distribution. With a parameter matrix $M = \left\{m_{ij}^a\right\}$, the joint density is given by

$$f(P|M) = \eta(M) \prod_{i,j=1}^{N} \prod_{a=1}^{A_i} \left(p_{ij}^a\right)^{m_{ij}^a - 1},$$

where $\eta(M)$ is a normalizing constant.

In this case, the Bayes update of distribution parameters in light of observation takes on a particularly simple form:

$$M' = T_{ij}^a(M) = M + \delta_{ij}^a.$$

Here we identify $\psi$ with the $A \times N$ matrix, $M$, of Dirichlet parameters, and $\delta_{ij}^a$ with a $A \times N$ matrix with a "1" in the position associated with the observation (*i.e.*, row $\sum_{l=1}^{i-1} A_l + a$, column $j$) and all other elements are zero. Note that, by the nature of Bayesian updating of $M$, it is equal to the initial matrix of parameters defining the prior, "offset" by the number of observed transitions; that is, with $M_o$ denoting the initial prior parameter matrix, and $F$ denoting a matrix of observed transition frequency counts (*i.e.*, $f_{ij}^a$ counts the number of observed (state $i$, action $a$) $\rightarrow$state $j$ transitions),

$$M = M_0 + F.$$

### 1.3.3 Bellman equation

We may regard $(i, M)$ as a generalized state, or "hyperstate," where $i$ is the physical state of the Markov chain and $M$ indexes our uncertainty in $P$. Let $V_i(M)$ be the maximum over all policies (which map generalized states to actions) of the discounted infinite horizon sum of rewards starting in hyperstate $(i, M)$.

Given a transition from $i$ to $j$ under action $a$, the maximum of the posterior discounted reward is

$$r_{ij}^a + \gamma V_j\left(T_{ij}^a(M)\right).$$

The probability of this transition given action $a$, with respect to the prior, is

$$\bar{p}_{ij}^a(M) \equiv \int_P p_{ij}^a dH(P|M).$$

(If $H$ is taken to be a Dirichlet distribution with parameter $M$, then it can be shown that $\bar{p}_{ij}^a(M) = \frac{m_{ij}^a}{\sum_j m_{ij}^a}$.)

The expected immediate reward given action $a$, with respect to the prior is

$$\bar{r}_i^a(M) = \sum_j \bar{p}_{ij}^a(M) r_{ij}^a,$$

and Bellman's equation may be written as

$$V_i(M) = \max_a \left\{ \bar{r}_i^a(M) + \gamma \sum_j \bar{p}_{ij}^a(M) V_j \left( T_{ij}^a(M) \right) \right\}.$$

As with standard MDP's, Bellman's equation could serve as a basis for a successive approximation scheme for computing the optimal value function:

$$V_i^t(M) = \max_a \left\{ \bar{r}_i^a(M) + \gamma \sum_j \bar{p}_{ij}^a(M) V_j^{t-1} \left( T_{ij}^a(M) \right) \right\}.$$

Note that $V^t(M)$ is not computed with respect to $V^{t-1}(M)$, but rather with respect to $V^{t-1}\left(T(M)\right)$.

A computational approach for finite horizon problems (or a receding horizon approach for infinite horizon problems) could proceed by evaluating all terminal values at the terminal horizon. The successive approximation equation above could then sweep backwards to the initial hyperstate. The number of terminal hyperstates grows exponentially with the horizon, however, and requires a complicated indexing scheme to be efficient. Alternatively, one could consider a recursive implementation (that is, an implementation involving recursive function calls, with intermediate values stored in push-down stacks) for which space requirements grow only linearly with the horizon, though this is at the expense of greatly-increased time requirements due to the duplication of effort in computing intermediate values (that is, a recursive approach duplicates value function computations for hyperstate nodes reachable from the initial hyperstate via multiple paths—see Section 1.5.1 and Table 1.1).

## 1.4 What would a solution mean?

Suppose that somehow we could compute the optimal policy for the hyperstate process, which we are referring to as a Bayes-adaptive Markov decision process (BAMDP). This optimal policy is optimal with respect to the prior distribution, which describes our initial uncertainty about the process parameters (transition probabilities). One way of thinking about the computational procedures that we later propose is that they perform an offline computation of an online, adaptive machine. We may regard the process of approximating an optimal policy for the BAMDP as "compiling" an optimal learning strategy, which can then be "loaded" into an agent. The agent can be released into its environment, and with actions dictated by the compiled policy, the agent then behaves in a manner that is optimal with respect to its prior, which describes the distribution of environmental scenarios the agent is likely to encounter.

If one were to observe the agent in operation, its physical-state-to-action mapping would appear to be nonstationary; the agent would appear to adapt as it observes the consequences of its actions, but this adaptation is illusory in the sense that the agent is merely following its compiled policy, which is a static, nonstationary mapping from hyperstates to actions.

stationary

The performance of the compiled policy is bounded above by the performance of a "clairvoyant" strategy, in which we could consult an oracle to determine the true state of nature and a priori adopt the appropriate optimal policy. We can express the value of the clairvoyant policy abstractly as $\int_P \max_\pi V(\pi, P)dH(P)$, where $\pi$ denotes policy and $H$ is the prior distribution for $P$, the generalized transition matrix for all states and actions. A lower bound on the performance of the compiled policy is provided by the value of the best "reactive" policy, which is the best static, physical-state-to-action mapping with respect to the prior; i.e., $\max_\pi \int_P V(\pi, P)dH(P)$. What Howard refers to as the "value of (perfect) information" (Howard, 1966) may be regarded as the difference between the value of the clairvoyant policy, which is not practically realizable, and the value of the compiled policy defined over hyperstates.

## 1.5 Dynamic programming for a simple example

Consider again the two-state, two-action MDP (depicted in Figure 1.2) with unknown transition probabilities, whose corresponding hyperstate process is described by the BAMDP presented in Figure 1.6. We shall show how to compute the optimal policy for this process using conventional dynamic programming (value iteration).

Perhaps the most obvious characteristic of the BAMDP transition diagram is that it has an acyclic, "pyramidal" structure. With two feasible actions at each decision stage and two possible physical state successors for each of the actions, one may speculate that the transition diagram has a branching factor of 4; the transition diagram has $4^{depth}$ hyperstates at a given depth. For a horizon of 10, this suggests roughly one million hyperstates. This is Bellman's "menace of the expanding grid" (Bellman, 1961)—the number of hyperstates grows exponentially with the time-horizon.

Another feature of the hyperstate transition diagram is that the transition probabilities are known. Although the underlying transition probabilities between physical states are unknown, our Bayesian model (*a la* beta distribution parameters) leads to a hyperstate process with a known model.

Although in general the physical state component of hyperstate can change arbitrarily at each step, the information state component changes only slightly—one information state component increases by 1 at each step—the information state is a "pure birth" process. This means that (typically) our model for uncertainty is slowly and smoothly changing as we observe transitions.

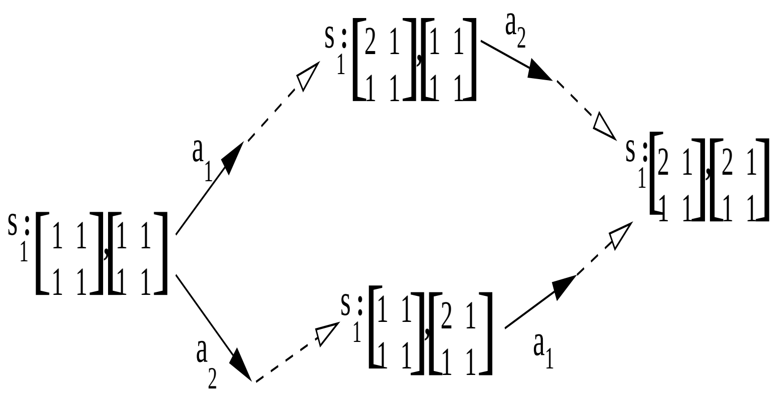

s_1: [1 1; 1 1], [1 1; 1 1]
a_1
a_2
s_1: [2 1; 1 1], [1 1; 1 1]
s_1: [1 1; 1 1], [2 1; 1 1]
a_2
a_1
s_1: [2 1; 1 1], [2 1; 1 1]

Figure 1.9: Multiple paths to a specific hyperstate.

### 1.5.1 How large is the hyperstate set really?

It is important that we take some care in selecting a data structure for representing the hyperstate set. Consider the trajectory that begins at Figure 1.6's initial hyperstate, applies action $a_1$, transitions into physical state $s_1$, then applies action $a_2$, and finally transitions into physical state $s_1$. The corresponding hyperstate at this point is $\left\langle s_1 \;\; ; \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \right\rangle$. But this is the same hyperstate that would have resulted if we had first chosen $a_2$, transitioned into physical state $s_1$, then chosen action $a_1$, and transitioned into physical state $s_1$ (see Figure 1.9).

Dynamic programming algorithms (and reinforcement learning algorithms) rely upon backups to improve their value function estimates, and it would be a mistake, or at least an inefficient use of computational resources, to represent these two identical hyperstates separately.

At a depth of 2, 15 of the 16 nodes are distinct—the hyperstate considered above is the only duplicate. However the number of duplicate nodes increases significantly as we consider greater depths. Table 1.1 compares the number of distinct nodes to the naive $4^{depth}$ estimate, and it can be seen that the difference is significant.

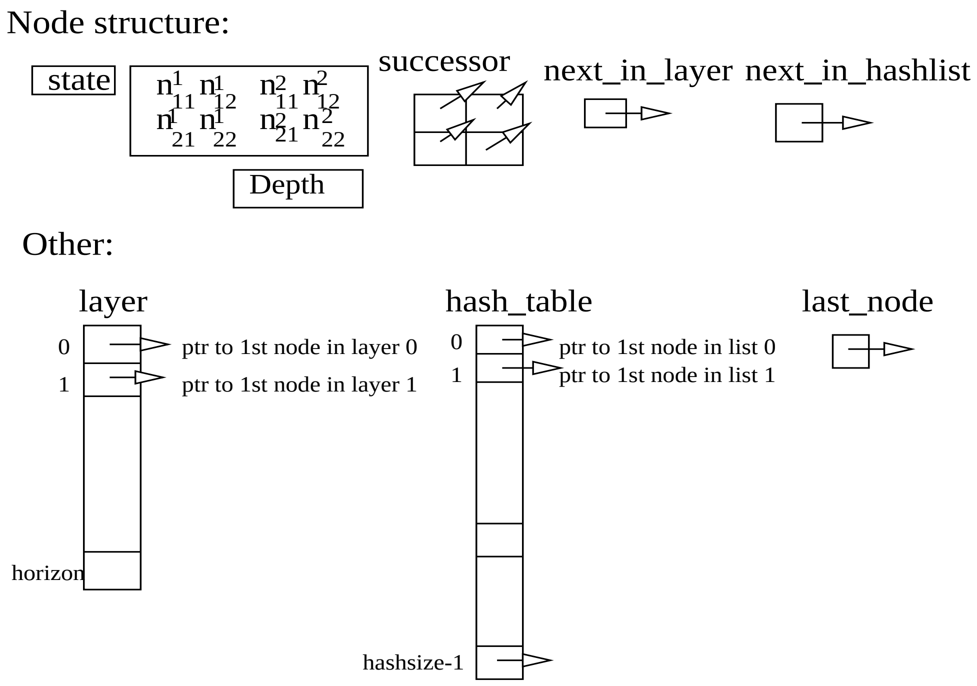
| depth | number of distinct nodes | $4^{depth}$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 4 |
| 2 | 15 | 16 |
| 3 | 46 | 64 |
| 4 | 118 | 256 |
| 5 | 264 | 1024 |
| 10 | 4625 | $1.04\times10^{6}$ |
| 15 | 30,748 | $1.07\times10^{9}$ |
| 20 | 128,094 | $1.10\times10^{12}$ |
| 25 | 403,702 | $1.12\times10^{15}$ |
| Total | 1,863,004 | $1.50\times10^{15}$ |

Table 1.1: Number of distinct hypertate nodes at various depths of the hyperstate transition diagram.

### 1.5.2 An efficient algorithm for enumerating distinct reachable hyperstates

The task of enumerating all distinct reachable hyperstates is a somewhat interesting problem in itself. Figure 1.10 presents the data structures that we find useful in accomplishing this task.

Our algorithm initializes the hyperstate space to be the initial hyperstate node, which exists at a depth of 0. We generate the various depth=1 action/next-state successors and add them to the space if they do not already exist. An array of pointers points to the first node in a linked list of nodes at each depth (one of the node structure fields points to the next node in the linked list). To generate all the distinct successor nodes at a given depth $d$, we generate all the possible successors from nodes at depth $d-1$. As each successor is generated, we check to see if it has already been generated. A key element of the algorithm is that hyperstate nodes are stored in a hash table, implemented as an array of linked lists. Newly discovered distinct nodes are inserted at the *head* of the list located at a position determined by the hash function (we view the information state as an eight-bit number in radix



Node structure:
state
$n^1_{11}$ $n^1_{12}$ $n^2_{11}$ $n^2_{12}$
$n^1_{21}$ $n^1_{22}$ $n^2_{21}$ $n^2_{22}$
Depth
successor
next_in_layer
next_in_hashlist
Other:
layer
0
1
ptr to 1st node in layer 0
ptr to 1st node in layer 1
horizon
hash_table
0
1
ptr to 1st node in list 0
ptr to 1st node in list 1
hashsize-1
last_node

Figure 1.10: Some useful data structures for enumeration of the hyperstate space.

HORIZON+1 notation then map this key to a hash table slot location using a simple modular function evaluated using Horner's rule—Figure 1.11 shows the resulting distribution of keys to hash-table slots). One of the node structure fields is the depth at which the node exists. To check if a given node pre-exists, we compute its hash value and traverse the corresponding linked list until it is found (it is a duplicate), we reach a node in the list with depth less than the depth of the searched-for node (it is distinct, and we can terminate the search since duplicate nodes must exist at the same depth and newly discovered distinct nodes are inserted at the head of the list), or we reach the end of the list (it is distinct). The resulting algorithm is reasonably efficient with respect to memory (memory is dynamically allocated as new distinct nodes are discovered) and time (linking nodes into a hash table greatly reduces the time required to check for the pre-existence of a given successor node).

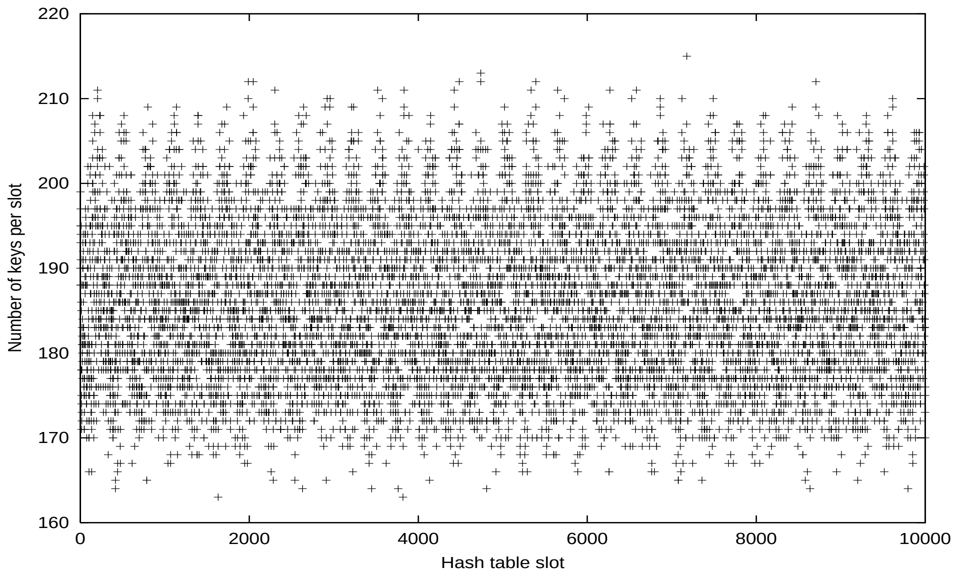


Figure 1.11: How the radix-(horizon+1) hash function distributes information state keys into hash table slots. The hash table size for this case was chosen to be 10007, and the keys were enumerated to a horizon of 25 (1,863,004 distinct information states).

### 1.5.3 Practical conclusions

In spite of special data structures introduced to help make the algorithm efficient, it cannot avoid Bellman's menace of the expanding grid, though since not all successor nodes at a given level are distinct, the effective branching factor is closer to 1.5 than it is to 4.

We shall take 30 steps as a practical, reasonable upper bound on the horizon that we can exhaustively enumerate and represent for this example.[6] Although 30 transition samples spread out over an information state with eight components works out roughly to only three or four samples per component, an exact solution to this problem will prove useful as a check for other less-exhaustive methods.

[6]By sprinkling the nodes randomly into hash table linked-list slots, the length of the average linked list is minimized, as is the time for search or insert—as long as the hash table fits in RAM. When the size of the hash table exceeds that of RAM however, we must swap in a page from virtual memory, and given that the hash function randomly distributes nodes into the hash table, memory accesses are random and so paging is frequently required. Ideally, we would like a data structure that makes use of the "locality of reference" properties possessed by the enumeration process, but preserves the fast search/insert time of the hash-table implementation.

### 1.5.4 Practical limits to classical methods: value iteration for BAMDP's

The fact that the hyperstate transition diagram is a directed acyclic graph (a special case of what Bertsekas and Tsitsiklis, 1996, call "sequential space decomposition") implies that we can compute the optimal value function using only one Gauss-Seidel iteration per state. That is, we can simply begin by computing the optimal 1-step value functions for nodes at LAYER= HORIZON-1, then compute the optimal 2-step value functions for nodes at LAYER= HORIZON-2, and so on until we compute the optimal value of the root. Bellman's equation takes the form:

$$V_i^t(M) = \max_a \sum_j \overline{p}_{ij}^a(M)[r_{ij}^a + V_j^{t-1}(T_{ij}^a(M)],$$

where $T$ transforms the prior information state, indexed by $M$, via a Bayesian update in light of transitions from state $i$ under action $a$—Bellman's equation dictates how values are to be backed up from the horizon sequentially back to the initial hyperstate.

### 1.5.5 A computational experiment

We add two new fields to our node structure in Figure 1.10 that store the optimal value function and optimal action. The existence of the layer array, whose elements point to the first node in each layer, and of the NEXT_IN_LAYER pointer field in the node structure, facilitates the process of sweeping through the various layers.

As an example, we computed the optimal policy for the two-state / two-action problem with the simple reward structure of $-1$ for landing in the left-hand state and $+1$ for landing in the right-hand state (see Figures 1.2 and 1.6). Prior uncertainty in transition probabilities is represented by beta distributions with parameters describing initial distributions that are uniform over all possible values. The initial physical state is taken to be the left-hand state. The optimal values for a selection of different horizons are presented in Table 1.2.

| horizon | $V^*$ |
|---|---|
| 10 | 1.499 |
| 15 | 2.816 |
| 20 | 4.236 |
| 25 | 5.719 |

Table 1.2: Optimal value for the initial hyperstate for our simple optimal learning problem.

The value function is a function of the information state, which resides in an eight-dimensional space. Figure 1.12 provides some indication of value function sensitivity with respect to information-state by plotting value as two parameters are varied: $\alpha_1^1$ and $\beta_1^1$ —which describe the prior uncertainty in transitions out of the left-hand state under action $a_1$ (other prior uncertainty distributions remain uniform). Intuitively, the optimal policy is one that maximizes the amount of time spent in the right-hand state, since its reward is greatest. Low values $\alpha_1^1$ of combined with high values for $\beta_1^1$ correspond to relatively high prior estimates for transitioning to the right-hand state under action $a_1$ compared with action $a_2$. For high values of $\alpha_1^1$ combined with low values for $\beta_1^1$ just the opposite is true.

## 1.6 Summary

This chapter began by presenting a number of examples of sequential decision-making under uncertainty, from drug administration (a bandit problem), to Markov decision processes with uncertain transition parameters. A characteristic of these examples is that optimal strategies may include probing actions, which gain information that may be utilized by actions taken in subsequent decision stages. We reviewed certain aspects of Bayesian statistics (the notion of conjugate families of distributions) and how the Bayesian perspective leads to a precise statement of the problem to be solved (and a precise characterization of its solution), but we also noted

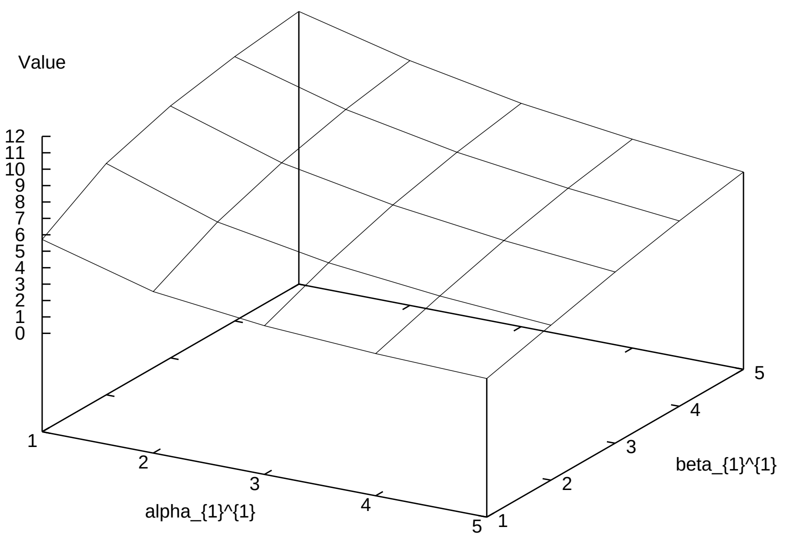


Figure 1.12: Sensitivity of optimal value to parameters describing prior uncertainty in transition probabilities from state 1 under action 1.

that conventional solution techniques entail levels of computation that are, in general, intractable, while seemingly reasonable heuristics, such as certainty-equivalence approaches, may exhibit performance that is arbitrarily bad.

We next presented an informal survey of reinforcement learning theory. Reinforcement learning algorithms are typically applied to Markov decision processes having extremely large numbers of states and possibly unknown dynamics. A variety of action-selection procedures have been proposed for ensuring asymptotic convergence to optimal policies (some of these procedures are presented in the following chapter), but these schemes do not directly address the problem of maximizing the cumulative reward obtained over the duration of the learning process.

Finally, we concluded with a relatively rigorous mathematical formulation of the optimal learning problem for Bayes-adaptive Markov decision processes, and we presented the details and limitations of a conventional dynamic programming approach for computing optimal learning policies.

# CHAPTER 2

# EXPLORE/EXPLOIT PERSPECTIVES AND PROCEDURES

A number of approaches for addressing the explore/exploit tradeoff have been proposed by researchers in the fields of adaptive control, decision theory, and reinforcement learning—this chapter provides a review.

## 2.1 Adaptive control processes and dual control

### 2.1.1 Bellman

Richard Bellman's concept of "adaptive control process" (Bellman, 1956, 1961; Bellman & Kalaba, 1959) has its origins in Wald's theory of sequential analysis in statistics (Wald, 1947). Before Wald, classical statistical tests operated in an open-loop manner; first, some number of samples were taken, then analysis was used to determine a decision. In Wald's sequential approach, the statistician could make use of the results of preceeding tests to determine the course of the testing program itself. Similarly, as an adaptive control process unfolds, additional information becomes available to the controller—the additional data may be accumulated as a result of deliberate probing, which itself is part of the control policy.[1] Bellman introduces the notion of "information pattern"—that information about the past history of the process which we wish to retain in order to guide our further actions. Using the theory of dynamic programming in a fashion similar to that employed in the study of deterministic and stochastic processes, he first presents a mathematical formulation of adaptive control processes in quite abstract terms. Then, incrementally, he shows how to reduce this level of abstraction to the point where he has constructive algorithms for the solution of numerical problems.

Bellman first postulates stochastic transformations for phase-space (physical-state space) and information pattern evolution in response to action. Appealing to the principle of optimality,[2] an optimality equation (recurrence relation) is derived for the expected value of a state/information-state pair. He next replaces the information pattern with a distribution function, then assumes that this distribution function is a member of a conjugate family, which allows the distribution function to be replaced by sufficient statistics. Bellman illustrates these ideas through the study of deterministic, stochastic, and adaptive versions of a linear control system and also through the study of the classic two-armed bandit problem.

### 2.1.2 Fel'dbaum

At about the same time as Bellman, in Moscow, Fel'dbaum (Fel'dbaum, 1965) initiated study of what he called problems of "dual control."

[1]"...A nonanticipative control cannot, obviously, use future observations that can benefit the performance of the control; however, the probabilistic description of these observations can reveal the 'value of future information' before they are actually taken. Therefore, while being non-anticipative, a control can still 'look into the future' and utilize what is presently known about the information to be obtained later. This is called 'preposterior analysis."' —(Bar-Shalom & Tse, 1976).

[2]"Preposterior analysis, which is 'anticipative' (in a statistical sense, *i.e.*, causal) of future information, is a consequence of the principle of optimality—the principle's statement that, at every stage, 'the remaining decisions must constitute an optimal policy with regard to the current information set' implies the following: every decision has to use the available 'hard' information (past measurements and controls) and 'soft' information (conditional densities) about the subsequent hard information. This can be paraphrased as the optimal controller has to know how to use what it knows as well as what it knows about what it shall know."—ibid.

Fel'dbaum proposed a taxonomy in which optimal systems are divided into three types:

(1) "Optimal systems with complete information about the controlled object or with maximal possible information;" these are deterministic control systems, or stochastic systems with probabilistic components described completely—they are amenable to classical variational methods, the maximum principle, or dynamic programming.

(2) "Optimal systems with partial information about the object and its independent (or 'passive') storage in the control process;" these are problems of optimal filtering, information theory and coding, and certain problems of statistical decision theory. A defining characteristic of these systems is that information accumulated over the course of time does not depend upon the actions of the controller—information is observed passively.

(3) "Optimal systems with partial information about the object and its active storage in the control process (dual control)."

In (3), the control law must acknowledge the interaction between the two controller functions of estimation and control. One form of interaction is exhibited by "non-separable" systems, in which the optimal control law requires information about the *accuracy* of estimates (not just their mean values). In the presence of uncertainty, there is usually a deterioration of system performance, which can be gauged by the decrease in value compared to the deterministic case. In order to reduce this decrease, the controller will tend to be risk averse, or "cautious." Another form of interaction is exhibited by "non-neutral" systems, where the control law influences the rate of reduction of uncertainty. This aspect may lead the controller to carry out what Fel'dbaum calls "active information gathering" or "probing," in which the controller introduces intentional errors in early stages of control so as to reduce uncertainty and improve performance over subsequent stages. Thus, control actions have a reciprocal character: "to a certain extent they must be of a probing nature, but also controlling to a known degree." This tradeoff between the objectives of parameter estimation and system control is referred to as *dual control.*

The duality of control is the fundamental physical fact distinguishing the third group of theories from the first two. In the first, dual control is not necessary, since without it the controller has complete information about the object. In the second, dual control is impossible, because information is accumulated by means of passive observation—its rate of storage does not depend at all on the strategy of the controller.

Fel'dbaum, like Bellman, also adopts a Bayesian perspective, and he suggests dynamic programming as a method for minimizing risk, accounting for evolving a posteriori densities for unknown parameters. Motivated by certain problems of communications theory, he examines a number of very simple ("inertialess") discrete-time linear and nonlinear feedback systems with noise processes injected into various points. Those examples turn out to be what Fel'dbaum calls "neutral systems"—their rate of information storage is independent of the control action. He extrapolates his approach to continuous time versions of these examples.

Non-neutral systems pose problems of storage, integration, and minimization that are "unrealizable in practice for large [horizon], even in the largest of modern [circa 1960] digital computers." Fel'dbaum notes however that in many cases, sufficient statistics can be used to ease the computational burden, then proceeds to apply his approach to "inertial" objects and to allow for more general random processes.

### 2.1.3 Caution and probing

Jacobs and Patchell (1972) further elaborate on the fact that a stochastic controller performs dual simultaneous functions. The interaction between the (conflicting) objectives of reducing uncertainty and exercising control can be framed in terms of the notions of "neutrality," "separability," and "certainty-equivalence."

*Neutrality*—A controlled object is neutral if the rate of reduction of uncertainty about uncertain variables is independent of the action; there is thus no reason to introduce a probing action.

*Separability*—Only expected values of uncertain variables are needed for the control law; information about the variance, or accuracy of conditional mean values as estimates of uncertain variables is not required.

*Certainty equivalence*—A certainty equivalence control law ($u^{CE}$) is the control law that would be optimal for the system if all random variations were replaced by their expected values. Systems with linear dynamics, additive Gaussian noise, and quadratic cost criteria have optimal controllers that are certainty equivalent. Certainty-equivalence implies that there is no interaction between the two functions of the controller; the control law can be designed without regard to stochastic effects.

These systems are related as follows:

- If the dynamics are linear in state, and noise is additive, then the system is neutral.
- If the system is neutral, and all random effects are normal, then the system is separable.
- If the system is separable, and the dynamics are also linear in the control, and the loss function is quadratic, then the system is certainty-equivalent.
- Certainty-equivalence implies separability.

Consider a simple example of a linear system with noisy observations:

$$\begin{aligned} x(k+1) &= x(k) + bu(k) \\ y(k) &= x(k) + z(k), \end{aligned}$$

where $x(0)$ and $z(k)$ are normally distributed with known, independent densities, and the performance criterion is to minimize $E\left[\sum_{k=1}^{N} x(k)^2\right]$. Bayes's rule yields a conditional probability function for the current state, given past observations and controls, that is normal with mean, $m_x(k)$, and variance, $\sigma_x^2(k)$, given by Kalman filter equations, and dynamic programming can be used to find the optimal control law: $u^*(k) = -m_x(k)/b$. For this example, $u^*$ depends only on $m_x(k)$ (the system is separable), and $\sigma_x^2(k)$ decreases independently of the control or observation sequence (the system is neutral). Also, $u^*$ is of the same form as the optimal control law, $u(k) = -x(k)/b$ for the system assuming no measurement noise and no uncertainty in state (certainty equivalence).

In contrast, consider the system (Kumar, 1983):

$$x(k+1) = x(k) + bu(k) + w(k),$$

with quadratic terminal cost $E\left(x(N)^2\right)$, where $b$ is unknown with an a priori distribution that is normal with mean $m_b(0)$ and variance $\sigma_b^2(0)$, and $w(k)$ is zero mean Gaussian with variance $\sigma_w^2$ for all $k$.

For a horizon of $N = 1$, the optimal control is $u(0) = -\frac{m_b(0)}{m_b(0)^2+\sigma_b^2(0)}x(0)$ with an associated optimal cost of $\frac{\sigma_b^2(0)}{m_b(0)^2+\sigma_b^2(0)}x(0)^2 + \sigma_w^2$. Note that as uncertainty in the unknown parameter $(\sigma_b^2(0))$ increases, the magnitude of the control signal decreases—it becomes cautious.

For a horizon of $N = 2$, the conditional optimal cost-to-go at $k = 1$, given the history or information set, $I(1) = (x(0), u(0), x(1))$, is given by the $N = 1$ horizon optimal cost evaluated using the conditional mean, $m_b(1) = E(b|I(1))$, and conditional variance $\sigma_b^2(1) = E\left((b - m_b(1))^2|I(1)\right)$; *i.e.*,

$$\frac{\sigma_b^2(1)}{m_b(1)^2 + \sigma_b^2(1)}x(1)^2 + \sigma_w^2.$$

An expression for $\sigma_b^2(1)$ may be obtained by viewing $x(1) = x(0) + bu(0) + w(0)$ as a noisy measurement of $b$ and by applying least-squares estimation theory:

$$\sigma_b^2(1) = \frac{\sigma_b^2(0)\sigma_w^2}{u(0)^2\sigma_b^2(0) + \sigma_w^2}.$$

To minimize the conditional cost-to-go, we desire that $\sigma_b^2(1)$ be small, and the formula for $\sigma_b^2(1)$ suggests that this may be achieved by making $|u(0)|$ large (to improve the "signal-to-noise ratio" in estimating $b$). However, from the state-dynamics equation, a large $|u(0)|$ tends to make $x(1)$ large, which *increases* the conditional cost-to-go. Dual control is concerned with the tradeoff between choosing a large, probing $|u(0)|$ to identify the unknown parameter, and a small, cautious $|u(0)|$ to avoid exacerbating the detrimental effects of uncertainty.

In the course of Jacobs and Patchell's investigations of these effects in the context of simple systems, they introduce some useful terminology:

*Intentional error*—The difference between a given control and the control generated by the feedback law that would be optimal if there were no uncertainty. Intentional errors arise precisely when there is an interaction between the two functions of a stochastic controller: reducing uncertainty and exercising control.

*Neutral control law*—The control law, $u^N$, that would be optimal in an equivalent neutral problem, where all interaction between control and future uncertainty is ignored.

*Probing*—$u^* - u^N$, where $u^*$ is the optimal control.

*Caution*—$u^N - u^{CE}$; uncertainty in a situation causes a cautious controller to act less strongly than if there were no uncertainty.

These definitions allow intentional error to be rewritten:

$$\begin{aligned} u^* - u^{CE} &= (u^* - u^N) + (u^N - u^{CE}) \\ &= \text{probing} + \text{caution}, \end{aligned}$$

or $u^* = u^{CE}+$ caution + probing, or $u^* = u^{CE}+$ intentional error.

The importance of these concepts derives from their suggestion of a structure for suboptimal control laws; that is, $\tilde{u}^* = u^{CE}$(controller designed by suboptimal estimates) + suboptimal caution and probing terms. For example, suboptimal probing may be comprised of test signals to maintain persistent excitation.

Bar-Shalom (1981) has proposed an approximate decomposition of the value function: $V = V^{CE} + V^{caution} + V^{probing}$, and noted that when uncertainty dominates, there are two cases: (1) $V^{caution}$ dominates—uncontrollable uncertainty; the model is highly-uncertain and cannot be improved in the course of the control period. (2) $V^{probing}$ dominates—which implies that one can use the dual effect of control to reduce uncertainty in the model. He provides an example that belongs to the second class, thus demonstrating that there are regimes in which dual control is a meaningful problem, contrary to Wonham's informal remark (Wonham, 1969) that, "in the case of (stochastic) feedback controls the general conclusion is that only marginal improvement can be obtained (over a controller ignoring the stochastic features), unless the disturbance level is very high; in this case the fractional improvement may be large but the system is useless anyway."

#### 2.1.3.1 Wide-sense dual control

Tse and Bar-Shalom (1973) apply their "wide-sense" dual control approach to problems with nonlinear dynamics/observation with additive noise. They make use of a quasi-sufficient statistic, the "wide-sense" information state (conditional mean and covariance), which is computed by filtering operations. They derive an optimality equation involving this information state, then expand the equation into a second-order Taylor series about a nominal trajectory (computed by Kalman filtering). This gives rise to a problem of minimizing quadratic cost subject to quadratic state equations, which is handled similarly to standard LQ problems. The solution requires solving a matrix Ricatti equation, and a boundary-value problem involving

Hessian matrices (that must be inverted) accounting for second-order perturbations. Their approach is computationally intensive particularly if the dimension of the controller is greater than one. In an example, their dual-controller performed roughly ten times better (in the sense of expected total reward) than a certainty-equivalence controller.

### 2.1.3.2 Suboptimal, practical strategies

A number of suboptimal, practical strategies have been proposed for addressing issues of dual control:

A common practice is to alternate between two pure modes of identification and control. In identification mode, the focus is solely on reducing uncertainty, which is often achieved by applying a pseudo-random sequence as a test signal, while in control mode, the certainty equivalent control is applied.

A myopic or "cautious" controller is basically a one-step ahead controller. There is no probing, and the controller becomes cautious when estimates are uncertain. This can lead to "turn off" phenomena, in which the control signal becomes very small. In practice, perturbation signals such as pseudo-random binary sequences or white noise are often added to cautious controllers to combat lack of excitation. The noise may be injected all of the time, or just when the variance of uncertain variables exceeds some level.

Constrained one-step minimization attempts to combat the turn-off phenomenon by incorporating constraints of the form, "limit the minimum value of the control signal." This may be done by placing a lower bound on the covariance matrix of estimates, but introduces application-dependent parameters.

One may consider certain extensions of the loss function. A two-step horizon controller leads to a minimization problem that has no analytic solution (though the loss function could be expanded to second order and minimized analytically).

Alternatively, one may add a term to the loss function that rewards good parameter estimates, a function of the covariance matrix of estimates, for example. This again leads to a minimization problem that must be solved numerically and that may have local minima.

## 2.2 Decision theory and the Bayesian control of Markov chains

By 1950 it was well recognized that the Markov chain is a useful model for a wide variety of physical processes, and an increasing number of applications of the mathematical theory have been made to problems in physics, chemistry, biology, and operations research (for example, see White, 1985, 1988). In these applications it is generally assumed that the matrix of transition probabilities is known, although questions of hypothesis testing and maximum likelihood estimation have been investigated (results are summarized by Billingsley, 1961).

During this time, Savage's interpretation (Savage, 1954) of the work of de Finetti on subjective probability rekindled interest in Bayesian decision theory. Contributions to this area were made by many researchers, including Von Neumann, Wald, Blackwell, and Girshick, leading to the work of Raiffa and Schlaifer (1961).

### 2.2.1 Silver

Research at MIT in the mid-to-late 1960's, primarily by students of Ronald Howard, applied Bayesian decision theory to various models based upon Markov chains with uncertain transition probabilities.

Silver's thesis (Silver, 1963) begins by considering the concept of a "multi-matrix" Markov process, in which it is assumed that one of several *known* transition matrices is being utilized—a Bayesian approach sequentially modifies a probability vector over the candidate matrices. Silver determines various quantities of interest, such as mean recurrence times, and examines certain aspects of decision making in the multi-matrix context.

Silver next considers the more general case where the transition probabilities are random variables. He shows that the Dirichlet (multidimensional Beta) distribution is the most convenient distribution, for Bayes calculations, to place over the probabilities of a single row of the transition matrix. He includes a method for assigning prior distributions, and examines the effect of randomly-distributed transition probabilities upon steady-state probabilities, first passage times, state occupancy times, and transient behavior. Silver considers decision problems, with particular emphasis on steady-state behavior, observation costs, and the value of information. Much of this work is generalized and made more rigorous by Martin (1967).

In the second half of Silver's thesis, he assumes that the transition probabilities are known exactly, but that rewards are random variables. He suggests appropriate conjugate prior distributions to use for the rewards when they are known to be drawn from a finite set or belong to certain ranges of real numbers. He considers the determination of expected rewards acquired during periods of transience or steady state, and provides guidelines for making statistical decisions for some special example scenarios.

### 2.2.2 Cozzolino et al.

Cozzolino, Gonzales-Zubieta, and Miller (1965) further explore Silver's Bayesian treatment of Markov decision processes with unknown transition probabilities. They propose a dynamic programming formulation for determining optimal strategies and carriy out the solution for a special two-state example, but conclude that applying their approach to non-trivial problems of any higher dimension is impractical. They then investigates heuristic methods for choosing sequential decisions (including a certainty-equivalence approach), and conduct Monte-Carlo simulation studies.

### 2.2.3 Martin

Martin (1967) rigorously obtains results under the assumption that the prior distribution function of the matrix of transition probabilities belongs to a family of distributions that is closed under consecutive sampling. He considers the discounted reward criterion and a successive approximation method for computing the solution to the corresponding set of functional equations (this approach is only viable for problems with a small number of states and short horizons). He also derives results for certain functions of the transition probabilities, such as the $n$-step transition probabilities and the steady state probabilities, and considers models in which the decision maker can sample various alternatives by paying a sampling cost. These results are stated in quite general terms, for any prior distribution function that belongs to a family closed under consecutive sampling. Martin then considers the special case where the distribution function is, what he calls, the "matrix beta" (or Dirichlet) distribution, a generalization of the beta conjugate family to the case of alternative multinomial sampling. He reviews and extends Whittle's (1955) distributional theory for state-transition frequencies, and specializes the results to the case of a two-state Markov chain, deriving moment-formulae, steady-state probabilities, and discounted value in terms of doubly-infinite (hypergeometric) series. This analysis is unlikely to generalize to problems with more than two states.

### 2.2.4 Satia

Satia (Satia, 1968; Satia & Lave, 1973) also considers Markovian decision processes with uncertain transition probabilities. First he considers a game-theoretic formulation, in which it is assumed that uncertainty is defined by upper- and lower-bounds on individual probabilities. Under the max-min criterion, nature selects probabilities to minimize expected total discounted total return, while the decision maker chooses a policy to maximize nature's min. Satia develops a policy improvement scheme that, in a finite number of iterations, finds a max-min policy whose value is within $\epsilon$ of optimal.

Next, Satia considers the Bayesian formulation in which uncertainty for transition probabilities is expressed in terms of conjugate probability distributions. Explicit enumeration of the decision tree corresponding to a sequence of alternative decisions is not practical but becomes feasible if partial enumeration and termination of some branches of the tree can be accomplished. Satia proposes a branch-and-bound technique, which maintains upper- and lower-bounds on the optimal discounted future return for each hyperstate node in the existing tree. (The bounds are derived by considering the value of max-min and min-max policies.) At any stage if the upper bound on the return for a particular hyperstate and decision, $d$, is less than the lower bound for some other decision, $d'$, then $d'$ "dominates" $d$ and the tree emanating from $d$ need not be considered further. Including successive stages in the decision tree results in refined bounds at the first stage, and if a decision is discovered that dominates all the other decisions at the first stage, then it is an optimal decision. The resulting algorithm is relatively complex and, while Satia's implicit-enumeration algorithm enhances the applicability of the Bayesian formulation (problems with four states and two decisions in each state can be solved), the required computational times are still excessive.

## 2.3 Action-selection heuristics for reinforcement learning

Q-learning's convergence theorem requires that all state-action pairs, $(s, a)$, be sampled infinitely often. One way of achieving this is through the use of "exploring starts;" *i.e.*, by requiring that there be a positive probability for starting each learning episode at each state-action pair. This may be of theoretical interest, and a viable alternative in the context of simulation, but it is unrealistic in situations where the agent must learn from real interaction with a real environment. For an agent interacting with and learning from its environment in an online way, we note that, in particular, simply following the greedy policy derived at each step from evolving Q-value estimates does not, in general, ensure convergence to an optimal policy. We would like our action-selection procedure to possess a sufficient exploratory component—to be of sufficient breadth to lead us eventually to an optimal policy—and yet we also desire action-selection to be efficient, to lead us to a good policy quickly, with good performance *en route*.

A number of heuristics have been proposed for action-selection that may be viewed as attempting to efficiently allocate exploratory actions to relevant portions of the environment. All nontrivial methods that follow involve parameters that influence the amount of exploration, and for many of the methods, these parameters can change through time. If parameters are tuned to ensure sufficient levels of exploration, then Q-values are guaranteed to converge to their optimal values. If, in addition, exploratory behavior is slowly phased out, then the policy will asymptotically align itself with the greedy policy, which will be optimal.

### 2.3.1 Uniform

The most basic and uninformed undirected exploration technique simply selects actions randomly with a uniform distribution, which one may suspect would lead to rather inefficient learning. Whitehead (1991) has considered the consequences of random-walk exploration in the context of deterministic "grid-world" environments with an explicit goal state, and has shown that in such settings every learning technique utilizing uniform-random exploration is expected to scale exponentially with the size of the state space. In contrast, he shows that if one adopts a simple counter-based action-selection rule ("go to the least-visited state") and has access to a predictive model, then the goal state is always found in time that is polynomial in the size of the state space.

### 2.3.2 Semi-uniform ($\epsilon$-greedy)

Given an agent in state $s$, we may define the "greedy" action as $\arg\max_a Q(s,a)$. In the $\epsilon$-greedy action-selection rule (*e.g.*, Sutton & Barto, 1998), with some small probability $\epsilon$, the agent chooses an action uniformly, while with probability $1-\epsilon$, it chooses the greedy action; that is,

$$Pr\left\{\pi_{Greedy}(s) = a\right\} = \begin{cases} 1-\epsilon+\frac{\epsilon}{|A(s)|} & a = \arg\max_{a'} Q(s,a') \\ \frac{\epsilon}{|A(s)|} & \text{otherwise} \end{cases}$$

where $|A(s)|$ denotes the number of admissible actions for state $s$.

### 2.3.3 Boltzmann

A widely-used action-selection mechanism, proposed by Watkins (1989), suggests choosing actions via the Boltzmann (or Gibbs) distribution:

$$Pr\left\{\pi_{Boltz}(s) = a\right\} = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}},$$

where the "Boltzmann temperature," $T$, is initialized to a relatively high value, resulting in a uniform distribution for prospective actions. As computation proceeds, temperature is gradually lowered, in effect raising the probability of selecting actions with higher Q-values; in the limit, the action that is greedy with respect to Q is selected.

One representative scheme (Barto, Bradtke, & Singh, 1995) for controlling the Boltzmann temperature cooling rate suggests updating $T$ via:

$$\begin{aligned} T(0) &= T_{max} \\ T(k+1) &= T_{min} + \beta\left(T(k) - T_{min}\right), \end{aligned}$$

where $k$ is the Markov chain step number, and $\beta$, $T_{min}$, $T_{max}$ are cooling rate parameters (example values are $\beta = .992$, $T_{min} = .5$, and $T_{max} = 75$).

### 2.3.4 Counter-based

Sato and Takeda (1988) present an action-selection scheme that occasionally deviates from a greedy strategy to execute actions that have been executed relatively infrequently. Their implementation maintains counts, $n_{ij}^k(t)$, of the number of transitions from state $i$ to state $j$ under action $k$ before time $t$. With $n_i^k(t) = \sum_j n_{ij}^k(t)$ denoting the marginal number of applications of action $k$ in state $i$, $\widehat{p}_{ij}^k(t) = \frac{n_{ij}^k(t)}{n_i^k(t)}$ forms the maximum-likelihood estimate of $p_{ij}^k(t)$, to which there corresponds a certainty-equivalent optimal policy, $\widehat{\pi^*}$,and value function, $\widehat{V^*}_i(t)$.

Given that the process at time $t$ is in state $i$, whose certainty-equivalent optimal action is $\widehat{a^*}_i$, Sato and Takeda's method executes an action specified by

$$\arg\max_k \left\{ \alpha I_{k=\widehat{a^*}_i} + \frac{c_i^k(t)}{n_i^k(t)} \right\},$$

where $I$ is the indicator function (equal to 1 if action $k$ is the certainty-equivalent optimal action), $c_i^k(t)$ gauges the infrequency of sampling action $k$ in state $i$, and $\alpha$ may be taken to be the certainty-equivalent value, $\widehat{V^*}_i(t)$, or may be regarded as a design parameter (constant). The second term within the scope of the arg max may be regarded as an "exploration-bonus" that biases action-selection toward actions that have been sampled infrequently.

If action $a$ is executed in state $i$ at time $t$, then Sato and Takeda propose updating the $c_i^k(t)$ via

$$c_i^k(t+1) = \begin{cases} c_i^k(t) + \theta\left(n_i(t+1) - n_i^k(t+1)\right) & k \neq a \\ c_i^k(t) & \text{otherwise,} \end{cases}$$

where $\{\theta(n)\}$, $n = 1, 2, \ldots$ is a bounded positive sequence that is either constant, or that is asymptotically zero with $\sum_{n=1}^{\infty} \theta(n) = \infty$ ($\theta(n) = \frac{1}{\sqrt{n}}$ yields good results in experiments). If this latter $\theta$-sequence is adopted, then Sato and Takeda prove that their action-selection procedure asymptotically converges to an optimal policy.

### 2.3.5 Recency-based

Motivated by a consideration of problems in which the environment is nonstationary, Sutton (1990) has proposed an action-selection procedure that uses counts, $n_s^a \; \forall s, a$, of the number of time-steps that have elapsed since action $a$ was executed in state $s$, and uses these counts to form an "exploration bonus" that is proportional to a heuristic measure of the Q-value's uncertainty. At each step, the agent selects the action that maximizes the Q-value plus its associated exploration bonus:

$$\pi(s) = \arg\max_{a'} \left\{ Q(s,a') + \epsilon\sqrt{n_s^{a'}} \right\},$$

where $\epsilon$ is some small constant and $\sqrt{n_s^{a'}}$ is the heuristic[3] scale-factor measure of the uncertainty in $Q(s,a')$. This approach is similar to Kaelbing's interval-estimation method, which is discussed in Section 2.4.1.

### 2.3.6 Mode switching

Cybenko, Gray, and Moizumi (1995) proposes interleaving "frames" of exploration and greedy control. In exploratory mode, the policy is randomized uniformly over all admissible actions. Let $N'$ be the maximum expected time for the process governed by this randomized policy to visit all states, starting from any state, and let $N = 2N'$ (these definitions rely upon certain recurrence and ergodicity assumptions). In the approach suggested by Cybenko et al., we apply the randomized policy until either (1) we obtain a sample transition for every Q-value state-action pair, or (2) we experience $N$ transitions under the randomized policy. If (1) occurs, we update the Q-values and apply the greedy policy with respect to the updated Q for $N$ transitions. If (2) occurs,

[3]The variance of the sample mean of an iid sample scales inversely with the number of samples; standard deviation scales inversely with the square-root.

we apply the greedy policy with respect to the previously estimated Q-values for $N$ transitions. The duration of subsequent greedy-policy frames increases by $N$ each time; the time spent operating in the exploratory mode becomes an asymptotically small fraction of the time spent operating under the estimated optimal policy, and asymptotic convergence to optimal Q-values and to an optimal policy is assured.

Thrun (1992) suggests defining separate error measures for the exploration problem and the exploitation problem. One can then choose actions to be optimized with respect to a static linear combination of the two criteria or an adaptive linear combination of the two criteria ("selective attention"). Thrun also proposes a scheme that dynamically switches between the two associated modes of action (pure exploration or pure exploitation) depending on their relative estimates of utility.

## 2.4 Principled heuristics

### 2.4.1 Interval estimation methods

#### 2.4.1.1 Bandit problems—IE

Consider a simple Bernoulli bandit problem (in which reinforcement is 1 or 0) with two arms (actions $a_1$ and $a_2$) with respective unknown success probabilities $p_1$ and $p_2$. The basic interval estimation approach (Kaelbling, 1993) maintains estimates of $p_1$ and $p_2$, along with information that allows us to construct confidence intervals for these estimates. The basic algorithm for action-selection chooses the action with the *highest upper bound* on expected reinforcement.

Let $n_i$, $i = 1, 2$, be the number of samples of arm $i$ and let the number of successes observed from these samples be $x_i$. A pair of $100(1-\alpha)$ % confidence intervals for the $p_i$ can be constructed using standard results from statistical theory; Kaelbling adopts the approximation (Larsen & Marx, 1986):

$$\mathit{UpperBound}(x,n) = \frac{\frac{x}{n} + \frac{z^2_{\alpha/2}}{2n} + \frac{z_{\alpha/2}}{\sqrt{n}}\sqrt{\left(\frac{x}{n}\right)\left(1-\frac{x}{n}\right) + \frac{z^2_{\alpha/2}}{4n}}}{1+\frac{z^2_{\alpha/2}}{n}},$$

where $z_{\alpha/2} > 0$ is defined to be the value that is exceeded by a standard normal variate with probability $\frac{\alpha}{2}$.

It follows that an action may be taken for two different reasons. First, the action's associated upper bound may be high because the observations suggest that the underlying $p$ is high, causing the entire confidence interval to be high. Second, the number of samples of the arm in question may be small, causing its confidence interval to be large. The parameter $\alpha$ essentially controls how exploration (acting to gain information) is to be balanced with exploitation (acting to gain reinforcement).

Kaelbling analyzes two sources of error for the interval estimation algorithm: "regular error" and "error due to sticking." Regular error occurs when the upper bound for the lower-probability arm is higher than the upper-bound for the higher-probability arm, in which case the algorithm samples the lower-probability arm repeatedly until its upper-bound is driven below the higher-probability arm upper-bound. Kaelbling derives an upper bound on the expected regular error for a finite-length run that is within a constant factor of optimal for the worst case.

If a sequence of samples causes the upper-bound for the higher-probability arm to move below the true probability of the lower-probability arm, then the algorithm may get stuck. The interval estimation procedure leads us to sample repeatedly from the lower-probability arm, whose confidence interval shrinks about its true payoff probability, while the higher-probability arm remains unsampled, its confidence interval remains fixed, in particular its upper-bound remains fixed at a level that directs us to sample the lower-probability arm. In this scenario, we are misled by a sequence of unlikely samples into a pessimistic perception of the higher-probability arm, and the interval estimation procedure does not direct us to resample and correct our estimate and its confidence interval. (It is possible, again through a unlikely sequence of samples, that the lower-probability arm confidence interval could shift down to an extent causing resampling of the higher-probability arm, but this is unlikely since the true lower-probability value is greater than the higher-probability upper-bound.) Kaelbling derives an approximation for the error due to sticking, which, together with the previous measure of regular error, provides an approximate upper bound on the total loss on a run of finite length as a function of the size of the confidence interval (which is parameterized by $z_{\alpha/2}$) and the distance separating the true payoff probabilities. Regular error increases with $z_{\alpha/2}$ while sticking error decreases. Prior information regarding the true payoff probabilities could guide us toward selection of a loss-minimizing value for $z_{\alpha/2}$; experimental results suggest using confidence intervals in the range of 95-99%.

Kaelbling's algorithm is closely related to Lai's algorithm (Lai, 1987), which can be shown to be optimal (regret-minimizing on runs of infinite length), but whose confidence bounds are more complex and not always tractably computable.

The preceding discussion has been within the context of Bernoulli bandit problems, bandit problems where the reinforcement is 1 or 0 with respective probabilities $p$ and $1 - p$. We may consider the more-general case in which reinforcement can take on real values. For example, if we presume that the reinforcement values are normally distributed, we can maintain first- and second-order statistics that allow us to form confidence intervals. For example, an upper bound of a $100(1 - \alpha)\%$ confidence interval for the mean is:

$$\text{UpperBound}_{\text{Normal}} = \frac{x}{n} + t_{\alpha/2}^{(n-1)} \frac{s}{\sqrt{n}},$$

where $s$ is the sample mean and $t_{\alpha/2}^{(n-1)}$ is Student's $t$ function with $n - 1$ degrees of freedom at the $\alpha/2$ confidence level.

Alternatively, one may adopt a nonparametric approach in which we store the actual observed reinforcement values (or such values from a sliding-window) rather than their statistics. Kaelbling notes that a procedure for deriving an upper bound of a $100(1 - \alpha)\%$ confidence interval for the distribution's center may be derived by appealing to the ordinary sign test (Gibbons, 1985): the upper-bound is provided by the $(n-m)th$ smallest element of the sorted data, where $n = \min$ {window width, number of instances received}, and where $m$ is defined to be the largest value such that

$$\sum_{k=0}^{m} \begin{pmatrix} n \\ k \end{pmatrix} .5^n \leq \frac{\alpha}{2}.$$

#### 2.4.1.2 MDP's—IEQ

The basic approach of interval estimation may be applied to the explore-exploit problem for MDP's. In this context, reinforcement (expected, discounted reward, for example) is "distal," or delayed; the value of taking a particular action in a particular state is function of immediate rewards obtained along the entire state trajectory, which is influenced by subsequent actions. We may regard value function estimates as appropriate, approximate measures of reinforcement and, as with interval estimation for bandit problems, we may construct confidence intervals for reinforcement (value) and define action-selection procedures based upon these intervals' upper bounds.

Kaelbling's IEQ procedure constructs confidence intervals about underlying Q-values and prescribes choosing the action with highest upper-bound. Q-values are, in general, real-valued estimates of distal reinforcement, and so the normal distribution model of reinforcement, or a non-parametric approach, may be used to construct the confidence intervals. In Q-learning, value functions are nonstationary, they track the value of the policy that is changing (improving) over time. Therefore, some mechanism is required for maintaining confidence intervals that bracket current values and disregard, or discount, the old. Non-parametric IE approaches that utilize a sliding-window address this problem directly. For the normal-distribution approach, we may decay the statistics, causing confidence intervals to grow, which may cause re-examination of actions whose values have been deemed inferior by value estimates corresponding to old policies.

Kaelbling applied IEQ to several example environments and concluded that "interval estimation techniques are not very satisfactory in the kinds of nonstationary environments that are encountered in learning from delayed reinforcement." Value function estimates can change rapidly as state transitions are observed, and this aspect poses a problem for IEQ.

#### 2.4.1.3 IEQL+

In the work of Meuleau and Bourgine (1999), the theory of bandit problems is used to define local measures of uncertainty, which are scaled and assimilated into immediate rewards, and then "back-propagated" via dynamic programming or temporal-difference mechanisms.

In the context of bandit processes, Gittins indices (discussed further in Chapter 3) may be computed and associated with each arm so that the optimal policy at each sampling step is to choose the arm with the largest index. Adopting normal models (unknown means and variances) for arm payoffs, the index, $\nu_i$, associated with arm $i$, is

$$\nu_i = \overline{r}_i + s_i \nu_g(0, 1, n_i),$$

where $\overline{r}_i$ is the sample mean of payoffs from arm $i$ (after $n_i$ samples), $s_i$ is arm $i$'s payoff sample variance, and $\nu_g(0, 1, n_i)$ is Gittins index of a normal arm with unknown and uniformly distributed mean and variance, after $n_i$ observations with sample mean 0 and sample standard deviation 1—these index values are provided in tabular form by Gittins (1989). Note that the index, $\nu_i$, has the same form as Kaelbling's upper-bound formula; in both cases, we are directed to choose the action which has the highest value of sample mean plus "exploration bonus," which estimates the value of acquiring information derived by sampling the arm.

In the context of MDP's, we may consider a similar approach in which Q-values play the role of arm payoffs; that is, for a process currently in state $i$, the agent is faced with a bandit problem where each action, $a$, corresponds to sampling a different arm, whose estimated payoffs are modeled by:

$$\rho_i^k = r_{ij}^k + \gamma \max_a Q_j^a \qquad w.p. \;\; p_{ij}^k.$$

Choosing a particular action causes a state transition at which point the agent faces the bandit problem associated with the new state. The transition also leads to an update of one of state $i$'s Q-values, which may affect the estimated payoffs associated with other states' arms. This is a violation of one of the fundamental assumptions of bandit theory: that bandit arms are independent of one another. Another violation of basic bandit theory assumptions is that not all arms are available for sampling at each decision epoch. In order to sample one of the arms associated with state $j$, for example, the agent must first control the MDP to state $j$.

Consider a version of Q-learning in which immediate reward is augmented by the exploration bonus:

$$N_i^k \stackrel{def}{=} (\overline{r}_i^k + \delta_i^k) + \gamma \sum_j \overline{p}_{ij}^k \max_a Q_j^a,$$

where $\delta_i^k$ signifies the exploration bonus, $s_i^k \nu_g(0,1,n_i^k)$, for example. This accounts only for the uncertainty associated with the current state. To acknowledge the uncertainty associated with all the future states, Meuleau and Bourgine consider a modified version of Bellman's equation:

$$\nu_i^k = (\overline{r}_i^k + \delta_i^k) + \gamma \sum_j \overline{p}_{ij}^k \max_a \nu_j^a.$$

Noting that the magnitude of the resulting criterion is of order

$$\sum_{t=0}^{\infty} \gamma^t (r+\delta) = V + \frac{\delta}{1-\gamma},$$

they suggest scaling the bonuses by $(1-\gamma)$, resulting in

$$\nu_i^k = (\overline{r}_i^k + \delta_i^k(1-\gamma)) + \gamma \sum_j \overline{p}_{ij}^k \max_a \nu_j^a,$$

a version of Bellman's equation whose solution can be computed using a modified version of Q-learning,

$$\Delta N_i^k = \alpha(n_i^k)\left[(r_{ij}^k + \delta_i^k(1-\gamma)) + \gamma \max_a N_j^a - N_i^k\right],$$

or other reinforcement learning algorithms. Meauleau and Bourgine argue that the $N$-values should be initialized to large positive values to ensure that the initial policy completely explores the environment, an instance of the "optimism in the face of uncertainty" heuristic. The final algorithm, IEQL+, proceeds similarly to Q-learning, but actions are prescribed by $\arg\max_k N_i^k$, and a sliding window of observations must be maintained for the computation of exploration bonuses.

A "worst-case" version of their algorithm uses prior bounds on rewards to form a worst-case measure of payoff variance, $\sigma_{max} \equiv \frac{r_{max}-r_{min}}{2(1-\gamma)}$, which is utilized in computing exploration bonuses. Adopting this technique eliminates the need for maintaining a sliding window of observations and appears to result in the best performance over three example tasks.

### 2.4.2 Model-based interval estimation

#### 2.4.2.1 MBIE

The interval estimation approach can also be combined with model-based reinforcement learning, rather than with direct methods, such as Q-learning. In research by Wiering and Schmidhuber (1998), maximum-likelihood estimates, $\overline{p}_{ij}^k$ and $\overline{r}_{ij}^k$, formed from STATE→ACTION→NEXT_STATE observations, define the MDP model. Dynamic programming techniques could be applied to this model. Wiering and Schmidhuber adopt a slightly-adapted version of prioritized sweeping (Moore & Atkeson, 1993) that maintains a priority-queue of states, where a state's priority is determined by the change in its value since it was last popped off the queue and processed. Each phase of the prioritized sweep involves: (1) Performing a model-based Q-value backup of the top priority state, $s$, (2) Performing a model-based Q-value backup of $s$'s predecessors, computing the values, $V_{s'} = \max_a Q^a_{s'}$, $s' \in Pred(s)$, of these predecessor states, and accumulating $\Delta(s')$, $s' \in Pred(s)$, the total change in value for each of the predecessor states since they were last removed from the queue and processed, and (3) Promoting the predecessor states to priority level $\Delta(s')$ in the priority queue.

MBIE replaces the model-based Q-value backup operations in prioritized sweeping with backups that use a modified version of the maximum likelihood model. For a given state-action pair, $(s, a)$, the transition probability for the "best" transition (as measured by $\overline{r}^a_{ss'} + \gamma V(s')$) is synthetically increased by an amount that is identical to Kaelbling's IE exploration bonus (in which the maximum-likelihood estimate for the best transition probability plays the role of IE's $\frac{x}{n}$). Other transitions are rescaled so that the modified transition probability matrix remains valid, and prioritized sweeping's model-based Q-value backups are performed using the modified transition model.

It is not entirely clear why prioritized sweeping should be modified in this way exactly, other than it seems, essentially, to be another instance of the "optimism in the face of uncertainty" heuristic. MBIE modifies the transition model so that the transition with highest estimated value has its probability elevated to its upper-bound. On a maze task, MBIE collected much more cumulative reward during training than simple frequency- or recency-based heuristics, "thereby effectively addressing the exploration/exploitation dilemma."

#### 2.4.2.2 Dirichlet-based MBIE

The transition probability upper-bounds utilized by MBIE are based upon a Gaussian model for uncertainty, which is valid for large sample sizes (Wiering and Schmidhuber actually utilize frequency-based action selection in the early stages of exploration, then switch over to MBIE). If one were to adopt a Bayesian approach, one would be led to consider Dirichlet distribution models for transition-probability uncertainty; this is the approach pursued by Wyatt (2001).

Wyatt addresses the problem of choosing a prior for the unknown transition probabilities, which is defined by a parameter matrix, $M$, by defining a single additional terminal state, $k$, to represent possible unobserved transitions. In the case where we have no prior knowledge about the transition matrix, $M$ consists solely of hypothesized transitions to $k$ from every state action pair; that is, the prior is described by a set of $m^a_{ik}$ for every state-action pair $(i, a)$. For states other than $k$, the value function is initialized to one-step expected rewards (assumed to be known), while $V_k$ is set to some value that upperbounds the value function. At each step, the agent, in state $s$ say, takes the greedy action, say $a$, with respect to MBIE's modified Q-value and observes a transition. If the transition has been experienced before, $M$ undergoes a standard Bayesian update. If the transition is a novel one, to $s'$, then $M$ essentially "grows" a new column entry in row $s$ to accommodate parameter $m^a_{ss'}$, which models uncertainty in the probability of the previously-unexperienced transition.

Wyatt's algorithm next uses the updated $M$ to calculate the upper bound of the $(1 - \alpha)100\%$ confidence interval for the transition probability to state $k$ for each action. This computation makes use of the fact that the marginal density required is a beta density; upper-bounds for a suitable parameter set can be computed offline. From this point, the algorithm proceeds similarly to MBIE, other probabilities are normalized, and the exploration-optimistic transition matrix is used with some form of asynchronous real time dynamic programming.

The initial set of prior $m^a_{ik}$ serve as parameters that affect exploratory behavior. High values cause high levels of exploration over a long period of time, while low values cause a rapid shift into an exploitatory mode.

Wyatt tests his algorithm on Wiering and Schmidhuber's maze task and shows that Dirichlet-MBIE outperforms (in terms of total and expected reward) both MBIE and a version of Meuleau and Bourgine's algorithm.

### 2.4.3 Bayesian Q-learning

Dearden's Bayesian Q-learning (Dearden, Friedman, & Russell, 1998) is an extension of conventional Q-learning that maintains and propagates probability distributions over the Q-values. The distributions are used to compute a "myopic" approximation to the value of perfect information for each action, and prescribes an action-selection procedure that balances exploration with exploitation.

Let $R_{s,a}$ denote the *total* discounted reward obtained starting from state $s$, executing action $a$, and thereafter taking actions prescribed by the optimal policy. $R_{s,a}$ is a random variable; its expected value is the (optimal) Q-value, $Q^*(s,a)$.

We are initially uncertain about the distribution of $R_{s,a}$, and Dearden makes several simplifying assumptions: (1) $R_{s,a}$ has a normal distribution, which is specified by its mean, $\mu_{s,a} = Q^*(s,a)$, and precision, $\tau_{s,a} = 1/\sigma^2_{s,a}$. (2) The prior, $p(\mu_{s,a}, \tau_{s,a})$, is a "normal-gamma" distribution, which is nonstandard terminology for the assumption that the prior conditional probability for the mean, given the precision, is normal, and that the prior marginal distribution for the precision is a gamma distribution. This is the appropriate conjugate family of distributions for sampling from a normal distribution with unknown mean and variance (see Degroot, 1970, Section 9.6). A set of four parameters specify a member of this family. (3) The prior over $\mu_{s,a}$ and $\tau_{s,a}$ is independent of the prior over $\mu_{s',a'}$ and $\tau_{s',a'}$ for $s \neq s'$ or $a \neq a'$. This is a restriction on the form of prior knowledge of the system. It is also assumed that, at any stage, the posterior over $\mu_{s,a}$ and $\tau_{s,a}$ is independent of the posterior over $\mu_{s',a'}$ and $\tau_{s',a'}$ for $s \neq s'$ or $a \neq a'$. This assumption is violated, in general, since $R_{s,a}$ for different $s, a$ are related through Bellman's equation.

Bayesian Q-learning differs from conventional Q-learning in that, for each state-action pair, $(s, a)$, we store the hyperparameters, denoted by $\rho_{s,a}$, which describe our uncertainty in the Q-value, $Q(s, a)$, rather than $Q(s, a)$ itself.

Dearden's "Myopic-VPI" action-selection procedure is motivated by certain aspects of the value of perfect information (Howard, 1966). Consider, for example, how knowledge of the true value of $\mu^*_{s,a}$ of $\mu_{s,a}$ might affect the agent's policy: (1) Suppose action $a_1$ is currently perceived as being the best action; that is, $E(\mu_{s,a_1}) \geq E(\mu_{s,a'})$ $\forall a'$. An oracle now reveals to us that, in fact, $a$ is a better action, $\mu^*_{s,a} \geq E(\mu_{s,a_1})$. A measure of the value of perfect information in this case is the value gained by knowledge of this fact, $\mu^*_{s,a} - E(\mu_{s,a_1})$. (2) Suppose that $a$ is currently perceived as being the best action and that $a_2$ is perceived as second best, and again suppose that an oracle reveals $\mu^*_{s,a}$ to us, but this time $\mu^*_{s,a} < E(\mu_{s,a_2})$; that is, in fact $a$ is not the best action, and the value gained by this knowledge is $E(\mu_{s,a_2}) - \mu^*_{s,a}$. Dearden thus defines the value gained from knowledge of $\mu^*_{s,a}$ :

$$Gain_{s,a}(\mu^*_{s,a}) \equiv \begin{cases} E(\mu_{s,a_2}) - \mu^*_{s,a} & \text{if } a = a_1 \text{ and } \mu^*_{s,a} < E(\mu_{s,a_2}) \\ \mu^*_{s,a} - E(\mu_{s,a_1}) & \text{if } a \neq a_1 \text{ and } \mu^*_{s,a} > E(\mu_{s,a_1}) \\ 0 & \text{otherwise} \end{cases}$$

where $a_1$ and $a_2$ are the actions with best and second best expected values, respectively. The agent does not, in fact, have access to an oracle that can provide $\mu^*_{s,a}$, and so we compute the expected gain given our prior beliefs:

$$VPI(s, a) \equiv \int_{-\infty}^{\infty} Gain_{s,a}(\mu) Pr\{\mu_{s,a} = \mu\} d\mu.$$

Taking into account that the prior for $\mu_{s,a}$ is specified in terms of the normal-gamma distribution, it is possible to reduce the VPI expression to an equation that involves the distribution function of $\mu_{s,a}$ (which may be computed efficiently):

$$VPI(s,a) = \begin{cases} c + (E(\mu_{s,a_2}) - E(\mu_{s,a_1}))\,Pr\{\mu_{s,a_1} < E(\mu_{s,a_2})\} & \text{if } a = a_1 \\ c + (E(\mu_{s,a}) - E(\mu_{s,a_1}))\,Pr\{\mu_{s,a} > E(\mu_{s,a_1})\} & \text{if } a \neq a_1 \end{cases}$$

where $c$ is an algebraic expression involving the normal-gamma parameters.

$VPI(s,a)$ provides an upper-bound on the myopic value of information induced by exploring action $a$; it is optimistic in the sense that by taking action $a$, we are not revealed $\mu^*_{s,a}$, rather only one more sample. The expected cost of taking action $a$ is the value sacrificed by deviating from the currently-perceived best action: $\max_{a'} E\left(Q(s,a') - E\left(Q(s,a)\right)\right.$. Choosing the action that maximizes the value of perfect information minus this cost is equivalent to choosing the action that maximizes

$$E\left(Q(s,a)\right) + VPI(s,a).$$

Upon applying a particular action and observing a transition, $s^a \rightarrow_r s'$, we need to update the Q-value distributions. This task is complicated by the fact Q-values are estimates of *total* discounted reward, whereas the state-transition observation is a sample of only the very first piece of this cumulative reward.

Dearden's "moment updating" approach begins by noting that $R_{s'}$, the discounted sum of rewards obtained starting from state $s'$ and following the current policy, is modeled by a normal-gamma distribution, whose first two moments can be computed using standard techniques. This implies that we can compute the first two moments of the "sample" $r + \gamma R_{s'}$, which may be used to update the hyperparameters, $\rho_{s,a}$, at state $s$. This update is optimistic in the sense that the mean of $R_{s,a}$ is updated by the *expected* sample, $r + \gamma E(R_{s'})$, and as a result the precision of the mean of $R_{s,a}$ increases too fast, leading to low levels of exploration and possible premature convergence to a suboptimal policy.

This problem can be avoided by weighting the posterior distribution updates of $R_{s,a}$ by the probability of observing different sample values of $R_t$ :

$$p^{mix}(\mu_{s,a}, \tau_{s,a}|r, s') = \int_{-\infty}^{\infty} p(\mu_{s,a}, \tau_{s,a}|r + \gamma x)p(R_{s'} = x)dx.$$

This "mixture update" results in a posterior that is, in general, not normal-gamma. Dearden proposes computing the best normal-gamma approximation by minimizing the KL-divergence (Cover & Thomas, 1991) from the true distribution. The resulting procedure requires numerical (integration) procedures for computing posterior hyperparameter values.

Dearden proves that, if all actions are tried infinitely often in each state, then the moment updating procedure's approximate Q-values will converge to the true values. If we use myopic VPI for action-selection then convergence is no longer assured. Dearden compares the performance of Bayesian Q learning to a number of other techniques (including Kaelbling's IEQ and Meuleau and Bourgine's IEQL+) on two simple Markov chain problems and a maze navigation task. Mixture updating combined with VPI action-selection results in best performance, though Dearden notes that his algorithm involves significantly more parameters (which are tuned for the experiments) than other methods.

### 2.4.4 Polynomial-time bound for reinforcement learning

In order for reinforcement learning algorithms to converge to optimal policies, their action-selection procedures must be of sufficient breadth; insufficient exploration can lead to convergence to sub-optimal behavior. Proofs of convergence typically stipulate that every action in every state must be sampled infinitely often, but simple learning schemes based upon this directive may exhibit slow convergence to optimal policies (convergence proofs are asymptotic in nature). Kearns and Singh (1998) propose an algorithm for learning near-optimal behavior with provably *poly-*

*nomial* (in the number of states and mixing-time of the optimal policy) bounds on actions and computation time. Operationally, their algorithm is another instance of mode-switching between exploration and certainty-equivalent, greedy exploitation.

Initially, while performing "balanced wandering" (taking the least-tried action in previously-visited states and arbitrary actions in newly-visited states), their algorithm maintains simple statistics for state-transition probabilities and expected rewards. During the wandering phase, states fall into one of three categories: (1) "known states," which have been visited sufficiently so that their transition probabilities and expected rewards are known with some degree of confidence, (2) "unknown states," which have been visited, but not to a sufficient extent to be categorized as known, and (3) "unvisited states."

The "known-state MDP," $M_S$, is defined as an MDP induced by the set of currently-known states, $S$. Transitions in the full MDP, $M$, between states in $S$ are preserved in $M_S$, while all other transitions in $M$ are redirected to a newly-defined absorbing state, $s_0$, representing unknown and unvisited states. Maximum likelihood estimates of transition probabilities and rewards for known states constitute a good approximation, $\hat{M}_S$, for $M_S$, an approximation that can provide good simulation accuracy for that part of $M$ that the algorithm knows well.

It can be shown that either the optimal policy is such that state-trajectories stay within $S$ with high probability or leave $S$ within the policy's mixing time, $T$.[4] Either case can be detected and replicated (by finding, respectively, an exploitation policy in $\hat{M}_S$, or an exploration policy that quickly drives $\hat{M}_S$ to state $s_0$). In somewhat more concrete terms, upon reaching a known state during balanced wandering, the algorithm performs value-iteration on $\hat{M}_S$, (the exploitation MDP) and on a version of $\hat{M}_S$, (the exploration MDP) in which all rewards are zero except for the reward, $R_{max}$, associated with state $s_0$. If the resulting exploitation policy's estimated value

[4]The "$\epsilon$-return mixing time" of a policy is the smallest time, $T$, such that the expected $T'$-step average return, for all $T' \geq T$, differs from the asymptotic average return by at most $\epsilon$.

is close to optimal, then this policy is executed for $T$ steps; otherwise, the algorithm executes $T$ steps of the exploitation policy.

Kearns and Singh's main result is a theorem stating that if the total number of actions and computation time taken by this algorithm exceeds a polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, N, T$, and $R_{max}$, then with probability at least $1 - \delta$, the policy computed by the algorithm has a value that is within $\epsilon$ of optimal.

Note that this theorem and its associated algorithm, as they stand, presume knowledge of the optimal value and of the optimal-policy's mixing time (these are inputs to the algorithm). These assumptions may be removed without violating the polynomial complexity bound, though the resulting procedure is impractical.

### 2.4.5 Summary

Bellman's Bayesian perspective for addressing sequential decision problems introduced the extremely valuable concepts of information pattern, the condensed representation of information pattern by conjugate families of distributions, and the dynamic programming approach for computing optimal decision policies. Fel'dbaum examined problems of "dual control" for dynamical systems, following an approach similar to Bellman's, and his perspective has been refined and extended by other control researchers.

Decision theorists have pursued Bellman's approach in the context of Bayesian Markov chains, developing a rigorous distributional theory for multi-step transition probabilities and steady-state distributions, though computing these quantities, or computing optimal policies for decision processes, remains an intractable proposition for even moderately-sized problems.

Reinforcement learning action-selection procedures have, for the most part, been driven by a desire to ensure asymptotic convergence to optimal polices, rather than by the desire to optimize cumulative reward over the duration of the learning process.

In this chapter, we surveyed a number of action-selection heuristics, some more principled (grounded in theory) than others. In particular, while researchers typically acknowledge the existence and computational intractability of Bellman's equation defined over hyperstates, it appears that none have attempted a direct assault, in which approximate dynamic programming methods are applied to Bellman's hyperstate system. We pursue this somewhat direct approach in Chapter 4. By retaining a complete world model that incorporates a Bayesian framework for characterizing evolving uncertainty, we derive policies that acknowledge long-term aspects of information gain and exhibit performance gains over simple heuristics. Moreover, in contrast to many heuristics, the justification or legitimacy of our policies follows directly from the fact that they are clearly motivated by a complete characterization of the underlying decision problem to be solved.

Before proceeding with this approach for general BAMDP's, however, we first examine a special class of problems of optimal learning whose optimal learning policies may be computed relatively efficiently by virtue of these problems' special structure.

# CHAPTER 3

# REINFORCEMENT LEARNING AND BANDIT PROCESSES

Multi-armed bandit problems are decompositionally-structured problems of optimal learning in which the hyperstate is composed solely of the information state—these problems constitute an important subclass of Bayes-adaptive Markov decision processes (BAMDP's), and have optimal learning strategies that can be computed relatively efficiently. A particularly elegant solution methodology was developed in the early 1970's by Gittins and Jones (Gittins, 1979, 1989; Gittins & Jones, 1974). Their approach casts the bandit problem as a collection of low-dimensional stopping problems (*i.e.*, sequential decision problems in which, at each stage, the agent must decide whether to stop with some terminal stopping reward, or to continue to make transitions and obtain process rewards), whose solutions determine the "Gittins indices," which in turn define the optimal policy.

This chapter begins by exploring the problem of learning the Gittins indices online without the aid of a process model; it suggests utilizing process-state-specific Q-learning agents to solve their respective stopping subproblems, and includes an example in which we apply our online reinforcement learning approach to a problem of stochastic scheduling—one instance drawn from a wide class of problems that may be formulated as bandit problems.

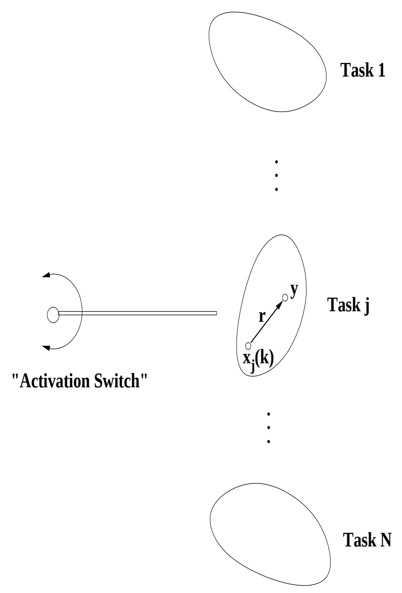


Figure 3.1: Bandit problem schematic.

In general, procedures for determining optimal adaptive controls for BAMDP's, optimal learning problems in which there *is* a physical-state component to the hyper-state, require a prohibitive amount of computation—the optimal learning problem for general BAMDP's is intractable. The final section of this chapter suggests an approximation approach in which bandit processes are used to model, in a certain "local" sense, a given BAMDP. Our approach proceeds by taking actions suggested by strategies that are optimal with respect to local bandit models.

## 3.1 Q-learning for multi-armed bandit problems

One basic version of the bandit problem concerns a collection of $N$ statistically independent reward processes and a decision-maker who, at each time $t = 1, 2, \ldots,$ selects one process to "activate." The activated process yields an immediate reward and then changes state; the other processes remain "frozen" in their current states and yield no reward (Figure 3.1). The decision-maker's goal is to splice together individual reward processes into one sequence of rewards having maximum expected discounted value.

Multi-armed bandits may be viewed as decompositionally-structured Markov decision processes. The size of the associated state sets may typically be of such magnitude as to overwhelm straightforward methods of solution. But these large state sets do possess a particular Cartesian-product structure and independence under various control actions, and one may exploit these defining characteristics. Proof that bandit problems can be decomposed into simpler, low-dimensional subproblems—in effect, rendering problems for which previous approaches had exponential complexity into problems with solutions of linear complexity—has been rigorously established by Gittins and Jones (Gittins, 1979, 1989; Gittins & Jones, 1974). Their approach reduces the problem of finding optimal policies for the original MDP to a sequence of low-dimensional stopping problems whose solutions determine the optimal policy through the "Gittins indices." Katehakis and Veinott (1987) have shown that the Gittins index for a process in state $i$ may be interpreted as a particular component of the maximum-value function associated with the "restart-in-$i$" process, a simple MDP to which standard solution methods for computing optimal policies, such as successive approximation, apply. The first part of this chapter shows how Q-learning can be integrated with this approach for computing Gittins indices to solve bandit problems online without assuming that the bandit process model is known.

After reviewing the bandit problem formulation, we gauge the complexity of computing optimal policies for a family of alternative bandit processes by modeling the family, straightforwardly, as one large Markov decision process. This is followed by a discussion of Gittins's approach, which is a comparatively efficient and elegant method with a number of interesting interpretations, one of which allows Q-learning to be applied.

Our central conceptual argument is summarized Section 3.1.4, in which we present the implementational details of a reinforcement learning algorithm for computing

Gittins indices. This is followed by several examples, as well as a discussion of important generalizations of the basic bandit formulation to cases of practical interest.

### 3.1.1 Multi-armed bandit problems

Suppose there exist $N$ stochastic processes $\{x_i(k)\}$, $i = 1, 2, ..., N$, whose values are members of a countable set. At each stage, $k$, a decision maker chooses an action, $a_k \in \mathcal{A} = \{1, 2, ..., N\}$.

Supposing that $a_k = j$, then the state, $\overrightarrow{x} = (x_1(k), \ldots, x_N(k))$, evolves according to

$$\begin{aligned} x_i(k+1) &= x_i(k) \quad i \neq j \\ x_j(k+1) &= f_j(x_j(k), w_j(k)), \end{aligned}$$

where $w_j(k)$ is a random disturbance depending on $x_j(k)$ but not on prior disturbances. For example, for Markov transitions, the state evolution is governed via $Pr\{x_j(k+1) = y\} = P_{x_j(k),y}$, where $P$ is a process state-transition matrix. This is the case considered henceforth.

The goal is to choose a sequence of actions $\{a_k\}$ to maximize the expected value of the discounted infinite horizon expected return:

$$\sum_{k=0}^{\infty} \gamma^k R_{a_k}\left(x_{a_k}(k)\right),$$

where $R_{a_k}(\cdot)$ is a bounded reward function and $\gamma \in (0,1)$ is the discount factor.[1]

Hence, the decision maker is essentially switching between the component processes' reward streams; activating a given process causes it to change state, while other component-process states remain "frozen."

[1]In some versions of this problem, at each time $k$, one also has the option of retiring permanently and receiving a one-time-only reward $\gamma^k M$. $M$ provides a useful parameterization for certain derivations or interpretations of the Gittins index, which will be reviewed in Section 3.1.2 (note that for $M$ sufficiently small, the retirement option can be excluded).

In the literature, each component process is referred to as a *bandit process*, while the entire collection of candidate processes is termed a *family of alternative bandit processes.*

The multi-armed bandit problem appears to have originated with the work of Thompson (1933). In a highly-influential paper, Robbins (1952) initiated systematic study of bandit problems that emphasized strategies for which asymptotic "loss" tends to zero. Bellman (1956) adopted a Bayesian formulation for the infinite-horizon discounted case, and Gittins and Jones (1976) generalized Bellman's process model to the one given above.

The term "multi-armed bandit" refers to the Bayesian adaptive control problem of selecting a sequence of plays on a slot machine that has several arms corresponding to different but unknown probability distributions of payoff. The formulation of a bandit process provided above is generalized somewhat from this classical Bernoulli-bandit process, depicted in Figure 3.2. (The drug allocation problem considered in Chapter 1 is also an example of a Bernoulli two-armed bandit problem, where arms correspond to alternative treatments.) For the Bernoulli-bandit case, bandit process states correspond to particular values taken on by sufficient statistics, which are related to the numbers of successes and failures observed for a given arm. Transition probabilities are the estimated probabilities of success and failure, which are sequentially updated using Bayes's rule. So, for the Bernoulli-bandit case, bandit process state-transition diagrams have an acyclic "fan-out" structure, with special values for transition probabilities, and all tasks have the same basic chain model (but with possibly differing priors). For special cases like this one, optimal allocation indices have been computed and are available in tabular form in books (*e.g.*, Gittins, 1989). However, the bandit process model we are adopting allows for arbitrary transition-graph topologies and general bandit process transition probabilities, which, for our case, may not even be assumed to be known.

trices are $|S|^N$-by-$|S|^N$, and standard methods for computing optimal policies have complexity of order roughly $O(|S|^{cN})$, where $c$ is a small constant. But non-activated processes do not change state, and rewards received depend only upon the state of the active process. These features may naturally lead one to conjecture the existence of decompositionally-defined optimal policies whose determination requires work on the order of only $O(N|S|^c)$. This is what researchers mean when they say, for example, that "...the multi-armed bandit problem was solved, after puzzling researchers for thirty-years, by Gittins and Jones" (Walrand, 1988).[2]

### 3.1.2 The Gittins index

So, what exactly are these Gittins indices?

The special Cartesian structure of the bandit problem turns out to imply that there are functions that map process-states to scalars (or "indices"), such that optimal policies consist simply of activating the process with largest index.

For example, at a given stage, each process in some state (see Figure 3.3(a)).

Associated with each of these component process states is a scalar index (Figure 3.3(b)). The optimal action is to activate the process whose current process-state has the largest index. The process-activation elicits an immediate reward and a state-transition (Figure 3.3(c)). There is another index associated with this new state, and it may no longer be the largest among all current process-state indices (in which case some other process will be activated at the next time-step).

Mathematically, the Gittins indices may be defined in the following way: Consider one of the reward processes, let $\mathcal{S}$ be its state space, and let $\mathcal{A}$ be the set of all subsets of $\mathcal{S}$. Suppose that $x(k)$ is the state of the process at time $k$ and, for $A \in \mathcal{A}$, let $\tau(A)$ be the number of transitions until the process first enters the set $A$. Let $g(i; A)$

[2]Whittle recalls (in the discussion following the paper by Gittins and Jones, 1979) that the efforts to solve the multi-armed bandit problem "...so sapped the energies and minds of Allied analysts that the suggestion was made that the problem be dropped over Germany as the ultimate instrument of intellectual sabotage."

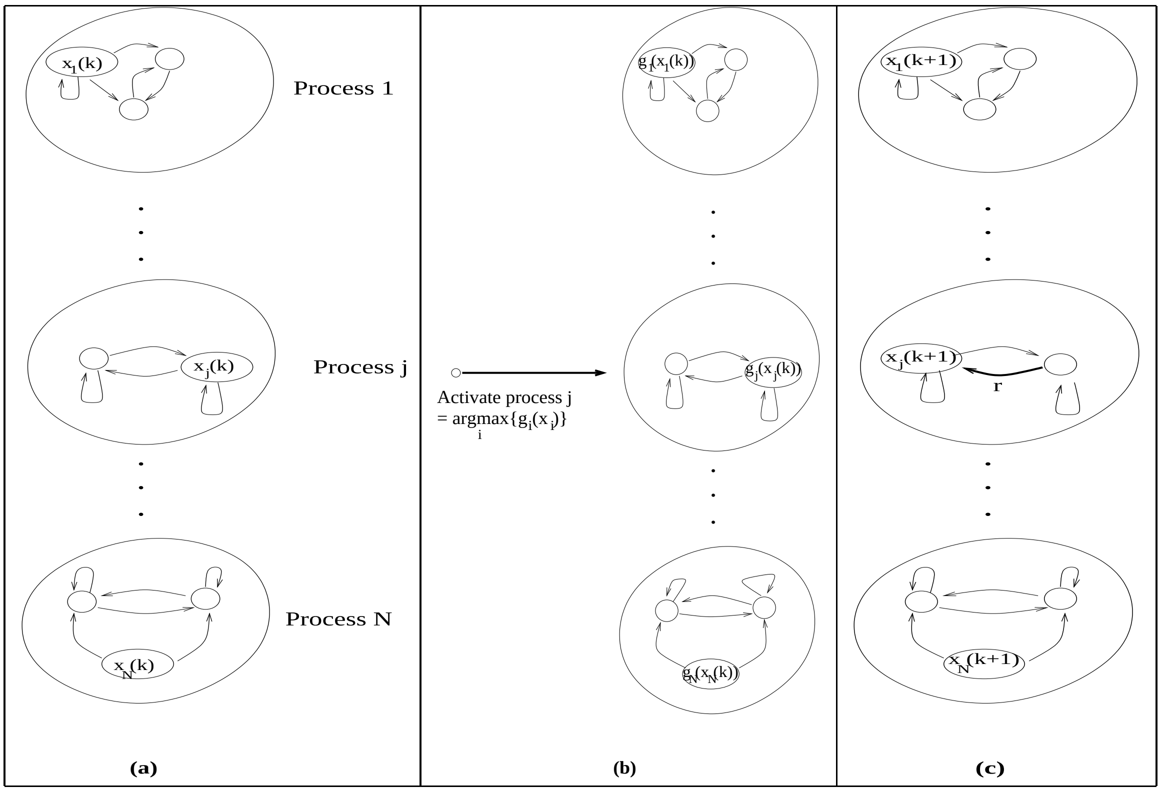


Figure 3.3: How the Gittins index determines optimal policies.

be the expected discounted reward per unit of discounted time starting from state $i$ until the stopping time $\tau(A)$:[3]

$$g(i;A) = \frac{E\left\{\sum_{k=0}^{\tau(A)-1}\gamma^k R(x(k))|x(0)=i\right\}}{E\left\{\sum_{k=0}^{\tau(A)-1}\gamma^k |x(0)=i\right\}}.$$

Then the Gittins index of state $i$ for the process in question is

$$g(i) = \max_{A\in\mathcal{A}} g(i;A). \tag{3.1}$$

Proofs that indices defined in this way determine optimal policies are notoriously difficult and obscure. Gittins and Jones's original proof (Gittins & Jones, 1974) used a sophisticated interchange argument. A different interchange argument was provided by Varaiya, Walrand and Buyukkoc (1985), and a simpler version was presented by Weiss (1988), who considered a more general branching bandits model.

[3]An integer-valued positive random variable $\tau$ is said to be a *stopping time* for the sequence $\{X(k)\}$ if the event $\{\tau = t\}$ is independent of $X(t+1), X(t+2), \ldots$ for all $t = 1, 2, \ldots$

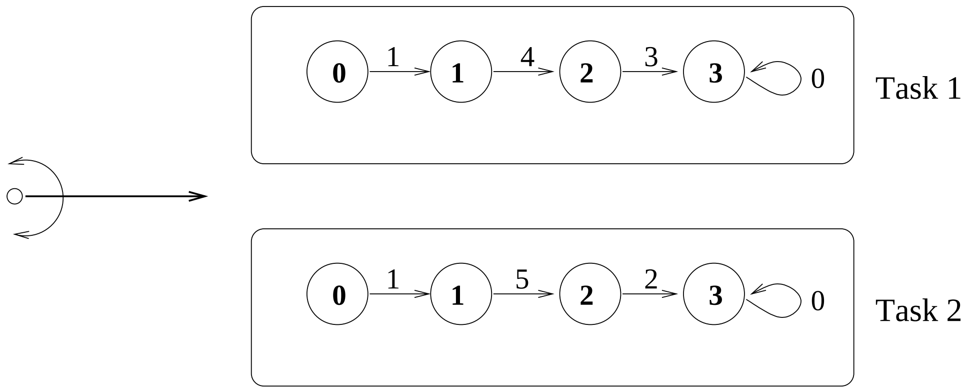


Figure 3.4: Merging problem modeled as a bandit problem (rewards label arcs).

Whittle (1980) provides an alternative proof based upon dynamic programming. Gittins (1979) shows that the indices may be computed by solving a set of functional equations. Other algorithms that have been suggested include those by Beale (see the discussion section following Gittins and Jones, 1979), Robinson (1981), Varaiya *et al.* (1985), and Katehakis and Veinott (1987).

#### 3.1.2.1 Example (Optimal merging of deterministic sequences)

Suppose we are presented with two sequences, or "stacks,"

$$\begin{aligned} x^1 &= \{1, 4, 3, 0, 0, \ldots\} \\ x^2 &= \{1, 5, 2, 0, 0, \ldots\}, \end{aligned}$$

which we wish to merge in such a way that the discounted ($\gamma = \frac{1}{2}$) sum of the resulting sequence is maximized. We may model this problem as a bandit problem in which the bandit processes have deterministic transitions (Figure 3.4). Let $x_i^k$ denote state $i$ of task $k$ (the number of times that bandit process $k$ has been played). The Gittins index is defined as:

$$g(x_i^k) = \max_{\tau > 0} \frac{\sum_{t=0}^{\tau-1} \gamma^t R(x_{i+t}^k)}{\sum_{t=0}^{\tau-1} \gamma^t},$$

where $R(x)$ is the reward associated with the arc leaving state $x$. For example, the

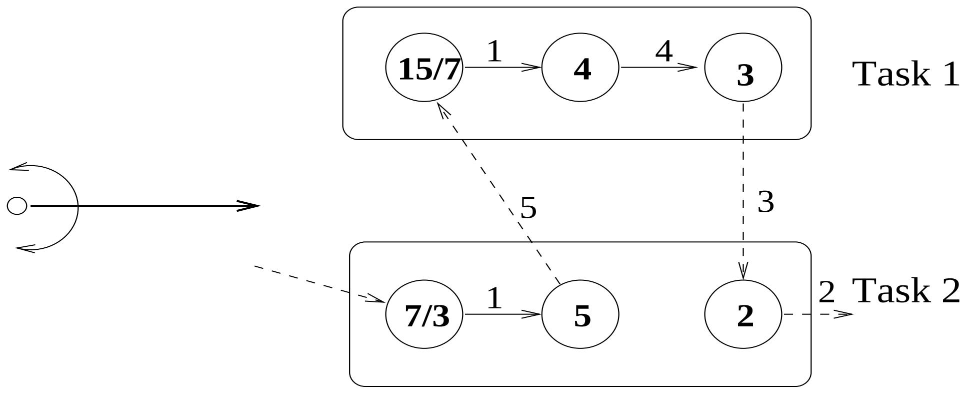


Figure 3.5: The optimally-merged sequence (Gittins indices label nodes).

index for $x_0^1$ is:

$$g(x_0^1) = \max\left\{1, \frac{1+\frac{1}{2}4}{1+\frac{1}{2}}, \frac{1+\frac{1}{2}4+\left(\frac{1}{2}\right)^2 3}{1+\frac{1}{2}+\left(\frac{1}{2}\right)^2}\right\} = \frac{15}{7}.$$

Other indices are indicated in Figure 3.5, which displays the optimal merge-ordering.

### 3.1.2.2 Interpretations of the Gittins index

Consider a version of the multi-armed bandit problem in which the decision maker has the added option at each stage $k$ of permanently retiring and receiving retirement reward $\gamma^k M$. Optimal policies have the form:

Retire if $M > \max_i\{g_i(x_i(k))\ \}$

Activate task $j$ if $g_j(x_j(k)) = \max_i\{g_i(x_i(k))\} \geq M,$.

where the Gittins index, $g_i(x_i(k))$, may be interpreted as an index of profitability for activating process $i$.

In order to gain insight into the meaning of the Gittins index, consider the bandit problem for a single process $i$. This is a standard stopping problem.

Let $V_i^*(x_i, M)$ be the optimal value function viewed as a *function of* $M$ for fixed $x_i$. For large values of $M$, $V_i^*(x_i, M) = M$, while for sufficiently small $M$, $V_i^*(x_i, M)$

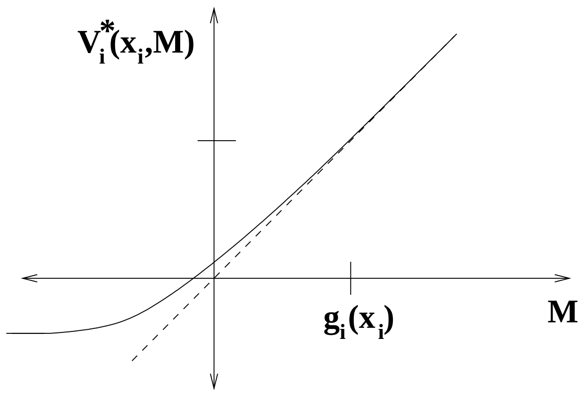


Figure 3.6: The Gittins index as an indifference threshold.

is some constant value independent of $M$ (*i.e.,* the optimal policy excludes retirement). Between these two extremes, it may be shown that $V_i^*(x_i, M)$ is convex and monotonically non-decreasing, and that there is a minimal value of $M$ such that $V_i^*(x_i, M) = M$ (Figure 3.6).

In fact, the Gittins index is this minimal value; that is, for all $x_i$,

$$g_i(x_i) = \min\{M | V_i^*(x_i, M) = M\}.$$

Thus, another interpretation for the index is that it provides an indifference threshold for each state between retiring and activating the process when in state $x_i$.

A related interpretation of the index may be derived (Ross, 1983) by again considering the single-process stopping problem in initial state $x_i$ and retirement reward $M = g_i(x_i)$; *i.e.,* the optimal policy is indifferent between continuing and retiring. It follows that, for any positive random retirement (stopping) time, $\tau$,

$$g_i(x_i) \geq E\left[\text{discounted return prior to } \tau\right] + g_i(x_i) E\left[\gamma^\tau\right], \tag{3.2}$$

with equality holding under the optimal continuation policy. Therefore,

$$g_i(x_i) = \max_{\tau > 0} \frac{E[\text{discounted return prior to } \tau]}{1 - E[\gamma^\tau]},$$

or

$$(1-\gamma)g_i(x_i) = \max_{\tau>0} \frac{E\left[\text{discounted return prior to } \tau\right]}{E\left[\text{discounted time prior to } \tau\right]}.$$

Thus, to calculate an index, it suffices to find the stopping time, $\tau$, such that the maximum reward per unit time (both discounted) prior to $\tau$ is maximal.

Weber (1992) provides an intuitive proof for the optimality of the Gittins index rule that is based on the notion of a fair game and an interpretation of the index that is equivalent to the previously-mentioned indifference-threshold interpretation. A similar view is presented by Ishikida and Varaiya (1994), who interpret the index as the winning bid in an auction for the right to use a "pipe," and by Gittins (1989), who calibrates candidate bandit processes against a "standard bandit process."

Suppose there is only one bandit process and that the decision-maker (gambler) may choose to activate (play) the process or not, but must pay a fixed *prevailing charge* for each play. For bandit $i$ in state $x_i$, one may define the *fair charge*, $g_i(x_i)$, as the value of prevailing charge for which optimal play of the bandit is a fair game; that is,

$$g_i(x_i) = \sup\left\{ g : \sup_{\pi} \left[ \sum_{t=0}^{\tau-1} \gamma^t \left(R_i(x_i(t)) - g\right) | x_i(0) = x_i \right] \geq 0 \right\},$$

where the stopping time $\tau$ is defined by the policy $\pi$.

As the bandit process state evolves, so too does the fair charge. In the event that the fair charge of the current state dips below the prevailing charge, in which case the gambler would normally stop playing, imagine that the prevailing charge is reset to the fair charge (Figure 3.7). Then the sequence of prevailing charges for each bandit process is non-increasing with the number of plays, and the gambler experiences continued play of a fair game. For the case of multiple bandit processes, by following a policy of playing the bandit of greatest prevailing charge, the gambler interleaves the prevailing charges from component bandit streams into one non-increasing sequence.

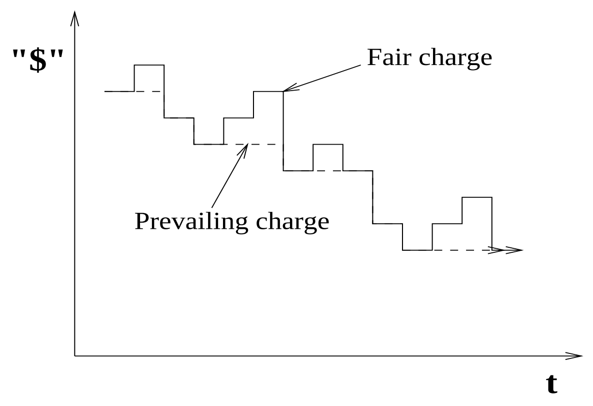


Figure 3.7: Fair charge and prevailing charge associated with an example bandit process trajectory.

By the nature of discounting, such a policy maximizes the expected total-discounted charge paid by the gambler. Since the gambler is engaged in playing a fair game, this policy maximizes expected total-discounted reward.

Whittle's (1982) proof that the index rule yields an optimal policy reveals along the way an interesting relationship that exists between the optimal value function of the overall multi-process bandit problem, $V^*(\overrightarrow{x}, M)$, and the optimal value functions of the component bandit processes, $V_i^*(x_i, M)$ :

$$\frac{\partial V^*(\overrightarrow{x}, M)}{\partial M} = \prod_{i=1}^{N} \frac{\partial V_i^*(x_i, M)}{\partial M}.$$

### 3.1.3 Restart-in-state-$i$ problems and the Gittins index

We have seen that optimal policies can be determined by calculating Gittins indices, and there are many ways of doing this. The method we shall make use of is due to Katehakis and Veinott (1987); it begins by considering the Markov-reward process associated with a *single* bandit process, and the "restart-in-state-i problem."

In this problem, for every state in the bandit process model's Markov chain, there are two admissible actions: CONTINUE and RESTART—see Figure 3.8.

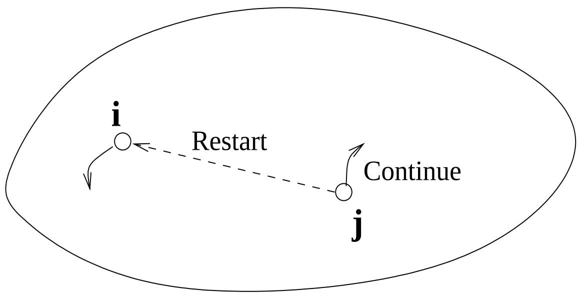


Figure 3.8: Restart-in-$i$ dynamics under CONTINUE and RESTART actions.

If the process is in state $j$ and the action is CONTINUE, then the process makes a state-transition and receives a reward just as it would if the process were activated and was in state $j$.

But if the action is RESTART, the process instantaneously teleports to the distinguished state $i$ (since we are considering the restart-in-$i$ process), and makes a state-transition and receives a reward from there as if the process had been activated while in state $i$.

If the decision maker's goal is to maximize the expected discounted return of this restart-in-state-$i$ decision process, then the dynamic programming equation for the optimal value function takes the form: for $j = 1$ to $n$,

$$V_j^i = \max\{\underbrace{r_j + \gamma \sum_k P_{jk} V_k^i}_{Continue},\ \underbrace{r_i + \gamma \sum_k P_{ik} V_k^i}_{Restart}\},$$

where $V_j^i$ signifies the $j^{th}$ component of the optimal value function for the restart-in-state-$i$ problem.

In particular, the $i^{th}$ component satisfies $V_i^i = r_i + \gamma \sum_k P_{ik} V_k^i$; that is, for the $i^{th}$ component, both terms over which we are maximizing are the same; the same thing happens regardless of whether the CONTINUE or RESTART action is taken (if the RESTART action is taken while in state $i$, then the process teleports to state $i$ and makes a CONTINUE-style transition from there), and so we can eliminate the "max" operator from this component of Bellman's equation.

The set of optimality equations may be written out explicitly, and solved via successive approximation:

$$\begin{aligned}
V_1 &\leftarrow \max\{r_1 + \gamma \sum_k P_{1k} V_k, V_i\} \\
V_2 &\leftarrow \max\{r_2 + \gamma \sum_k P_{2k} V_k, V_i\} \\
&\vdots \\
V_i &\leftarrow r_i + \gamma \sum_k P_{ik} V_k \\
&\vdots \\
V_n &\leftarrow \max\{r_n + \gamma \sum_k P_{nk} V_k, V_i\}
\end{aligned}$$

The main result, shown by Katehakis and Veinott, is that the $i$th component of the optimal value function for the restart-in-$i$ problem is the Gittins index for state $i$.

The main result is true for each of the states; that is, for each state $k$, there exists a restart-in-state-$k$ MDP yielding as the $k$th component of its optimal value function the Gittins index for state $k$.

For $N$ alternative bandit processes, each with $n$ states, this implies that one can compute all of the Gittins indices by computing optimal value functions for $Nn$ MDP's, each with two actions and $n$ states.

Recalling Ross's interpretation of the Gittins index, we note that $V_i^i$ may also be interpreted as the maximum value in state $i$ for the corresponding embedded single-state semi-Markov decision chain; that is, $V_i^i$ satisfies

$$V_i^i = \max_{\tau > 0} E\left\{[\text{discounted reward prior to } \tau] + \gamma^\tau V_i^i\right\},$$

where $\tau$ is a stopping time for the process, namely, the first period in which one chooses to restart in state $i$ in the restart-in-$i$ problem (see Figure 3.9). Comparing this last equation with Equation 3.2 in the Section 3.1.2.2 under an optimal contin-

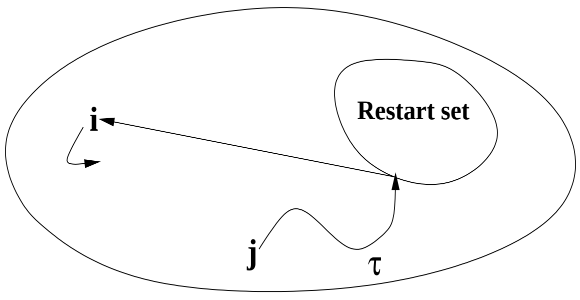


Figure 3.9: The optimal restart set for the restart-in-state-$i$ problem.

uation policy, one concludes that $V_i^i$ may be identified with the Gittins index, $g(i)$, for state $i$.

For a given state $i$, there exits a set of states, the "optimal restarting set," for which, once entered, it is optimal to restart in $i$. The number of transitions taken to reach this restarting set, starting from state $i$, is the optimal stopping time associated with the Gittins index.

### 3.1.4 On-line estimation of Gittins indices via Q-learning

The multi-armed bandit problem was stated in Section 3.1.1, and Section 3.1.2 presented the Gittins index approach for constructing optimal policies, an approach that reduces the bandit problem to a set of low-dimensional stopping problems; Section 3.1.3 asserted that the Gittins index for a given state may be characterized as a particular component of the optimal value function associated with a stopping problem, a simple MDP.

Reinforcement learning methods, such as Q-learning, are adaptive, model-free algorithms that can be applied online for computing optimal policies for MDP's. We recall from Chapter 1 that Q-learning (Watkins, 1989) was originally advanced as a sample-based, Monte-Carlo extension of successive approximation for solving Bellman's equation, and that alternative motivation and justification for the algorithm, as well as rigorous proofs of convergence, appeal to results from the theory of stochastic approximation—see Bertsekas and Tsitsiklis (1996).

1. For every state $k$ in this given process, there is an associated restart-in-state-$k$ process. And this simple $i \xrightarrow{r} j$ transition is consistent with taking the CONTINUE action in state $i$ for *all* $n$ of the restart-in-state-$k$ processes (see Figure 3.10(b)).

2. But it is just as valid to interpret this transition as being the result of taking the RESTART action in *any* state $k$ for the restart-in-$i$ process (since in that case we would instantaneously teleport to state $i$ and make a transition from there, see Figure 3.10(c)).

Assuming a bandit process with $n$ states, these overlapping, consistent interpretations of a single observed transition yield data that are relevant to *all* restart-processes associated with the given bandit process and, in effect, allow us to update $2n$ Q-values: for $k = 1, \ldots, n$,

$$Q(\text{state} = i,\ \text{action} = \mathit{Continue},\ \text{restart problem} = k,\ \text{bandit process}),\ \text{and}$$
$$Q(\text{state} = k,\ \text{action} = \mathit{Restart},\ \text{restart problem} = i,\ \text{bandit process}).$$

For $N$ alternative bandit processes and $n$ states per process, this implies a table of Q-values of size $2Nn^2$ (compared with size $Nn^N$ for a direct MDP approach). The current estimate for the Gittins index for a given state is given by the entry in the Q-table for which the state and restart-problem components match. Suppose that the collection of alternative bandit processes is in some given state $\vec{x} = (x_1(k), \ldots, x_N(k))$. The current estimate of the Gittins index for process $i$ in state $x_i$ is given by:

$$\tilde{g}_i(x_i) = Q(\text{STATE} = x_i,\ \text{ACTION} = \mathit{Continue},\ \text{RESTART PROBLEM} = x_i,\ \text{BANDIT PROCESS} = i).$$

It remains to define bandit process activations in a way that achieves an adequate sampling of states and actions. There are many reasonable ways of doing this; here we adopt a Boltzmann-distribution-based action-selection method: for $i = 1$ to $N$,

$$Pr\{\text{activate bandit process } i\} = \frac{e^{Q(x_i,C,x_i,i)/T}}{\sum_{i=1}^{N} e^{Q(x_i,C,x_i,i)/T}},$$

where $T$ is the "Boltzmann temperature," and "C" denotes the CONTINUE action. This is just the usual Boltzmann action-selection scheme, previously discussed in Section 2.3.3, but with Gittins index estimates playing the role of value estimates.

In summary, at each stage:

- Select a bandit process to activate via the Boltzmann distribution.
- Observe the state-transition $i \rightarrow j$ and immediate reward $r$ elicited by activating the process.
- Perform $2n$ backups, where $n$ is the number of states for the activated process: for $k = 1$ to $n$,

$$\begin{aligned}
&Q(\text{state} = i, \text{action} = \mathit{Continue}, \text{restart-problem} = k, \mathit{process}) = \\
&\qquad (1-\alpha) Q(\text{state} = i, \text{action} = \mathit{Continue}, \text{restart-problem} = k, \mathit{process}) \\
&\qquad +\alpha \left[ r + \gamma \max\nolimits_{a \in \{C,R\}} Q(\text{state} = j, \text{action} = a, \text{restart-problem} = k, \mathit{process} \right] \\
&Q(\text{state} = k, \text{action} = \mathit{Restart}, \text{restart-problem} = i, \mathit{process}) = \\
&\qquad (1-\alpha) Q(\text{state} = k, \text{Restart}, \text{restart-problem} = i, \mathit{process}) \\
&\qquad +\alpha \left[ r + \gamma \max\nolimits_{a \in \{C,R\}} Q(\text{state} = j, \text{action} = a, \text{restart-problem} = i, \mathit{process} \right]
\end{aligned}$$

—where "C" and "R" respectively denote the admissible actions, CONTINUE and RESTART.

If each of the $N$ alternative processes has $n$ possible states, then $2Nn^2$ Q-values must be calculated and stored. Note that this is a substantial reduction from the

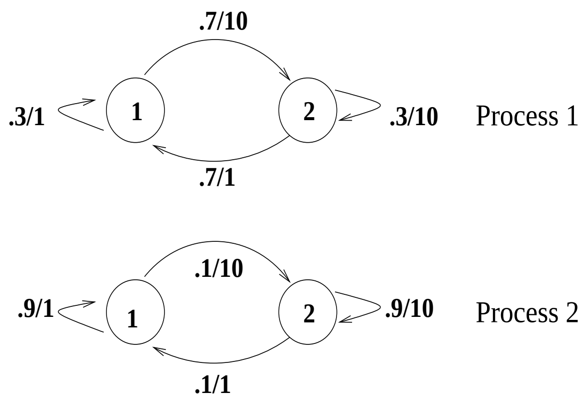


Figure 3.11: A simple bandit problem.

$Nn^N$ values required by an approach based upon the straightforward MDP formulation.

Moreover, each state-transition gives rise to $2n$ backups, and this effective parallelism may be viewed as further reducing the computational complexity. That is, to calculate all the Gittins indices, the algorithm solves $Nn$ MDP's (number of alternative bandit processes $\times$ number of restart-problems per process), each of size $n$. But for each process the associated $n$ restart-problems are solved in parallel, and are rather simple MDP's in that there are only two admissible actions per state.

A proof of convergence follows from existing convergence proofs of Q-learning for conventional MDP's (Bertsekas & Tsitsiklis, 1996), which assume adequate sampling of states and actions.

### 3.1.5 Examples

#### 3.1.5.1 A simple example

To confirm that this algorithm works, first consider the simple bandit problem shown in Figure 3.11.

This problem has two processes, each with two states. Transition probabilities/rewards label arcs, and the discount factor is chosen to be $\gamma = .7$. The optimal policy may be calculated by solving a four-state MDP, as discussed in Section 3.1.1,

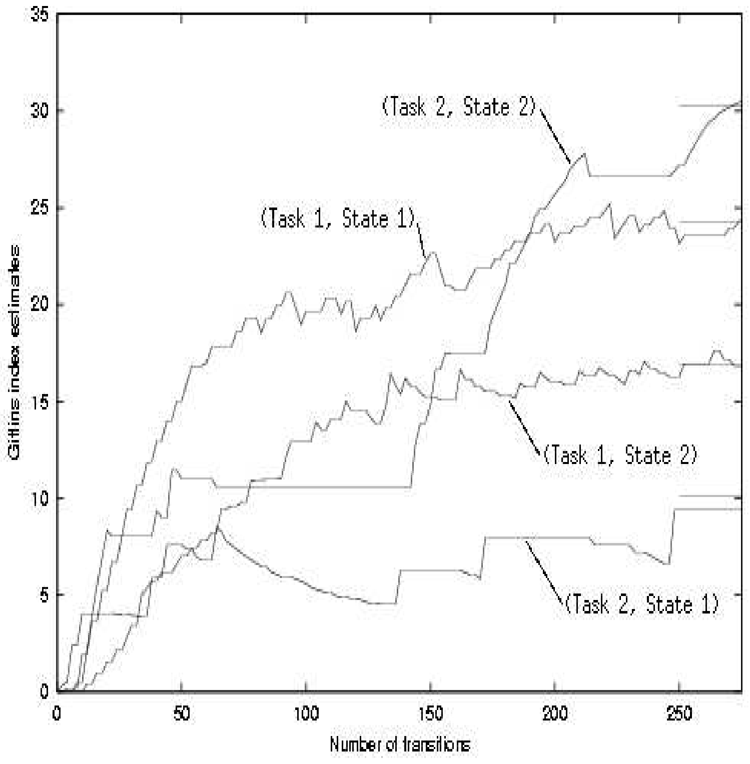


Figure 3.12: Convergence of Gittins indices for a simple example.

or by applying the model-based, successive approximation scheme of Katehakis and Veinott offline. The optimal policy is to activate PROCESS 2 if it is in STATE 2, but otherwise to activate PROCESS 1.

Figure 3.12 plots the convergence of the Gittins indices to their true values using our reinforcement learning algorithm. The optimal policy is to activate PROCESS 1 once PROCESS 2 leaves STATE 2. Consequently, as the Boltzmann temperature is lowered, an increasing number of transitions are made under the greedy policy with respect to the index estimates; that is, an increasing proportion of transition samples are drawn from PROCESS 1 activations. Miscellaneous parameters that govern the rate of Boltzmann temperature reduction have not been optimized; the purpose of this example has been simply to demonstrate that the on-line algorithm works.

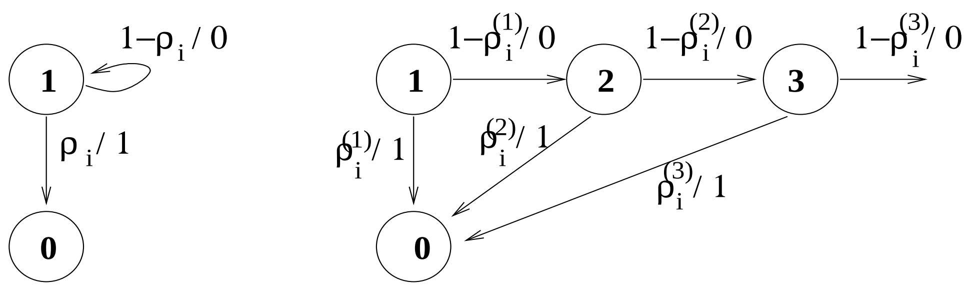
#### 3.1.5.2 Stochastic scheduling

A more meaningful example bandit problem is that of (static) stochastic scheduling. Consider the scenario in which there are some number of "sources" that generate "tasks," "jobs," "projects," or "customers" whose service requirements are described by respective distributions. In each trial, the "server" or "processor" is confronted with a batch of tasks from the sources. The server's goal is to allocate its processing time to tasks (possibly in a pre-emptive way) such that, say, the expected waiting time, averaged over tasks in a batch, is minimized. (The schematic at the bottom of Figure 3.14 is intended to convince the reader that the order of service indeed makes a difference in this respect.)

For example, consider the problem of task scheduling where each task $i$ has a geometric service time: $Pr\{\tau_i = s\} = \rho_i(1-\rho_i)^{s-1}$ (see Figure 3.13(a)). Each task $i$ thus has a constant "hazard rate," $\rho_i$, and it is known that, in order to minimize either mean flow time,[4] or mean waiting time, an optimal policy is to activate the tasks in decreasing order of this parameter. In a simple experiment, ten $\rho$'s were drawn uniformly from the unit interval, and the discount rate was set to $\gamma = .9$. We applied our reinforcement-learning algorithm (which assumes that the $\rho$'s are unknown), in a trial-based, online fashion (a trial consists of activating tasks in some manner until all tasks are completed). Index estimates converged to values whose magnitudes are consistent with the optimal highest-hazard-rate policy.

The constant hazard rate case was considered in this example because the optimal policy is known and provides a simple check. Non-constant hazard rate cases can be handled by defining the task models as suitable Markov reward chains, see Figure 3.13(b). Moreover, it may be unreasonable to presume that the service-time distributions are known *a priori*. In this case, our reinforcement-learning algorithm

[4] *Mean (weighted) flowtime* is defined as the (weighted) sum of task finishing times, divided by the number of tasks.



**(a)** **(b)**

Figure 3.13: Task models with (a) constant and (b) non-constant hazard rates.

can be applied, online, to calculate the respective Gittins indices directly, without building an explicit task model.

Let us consider a general version of the stochastic scheduling problem in which the service-time distributions are *not* assumed to be known. Rather, in a given trial, the server is confronted with a batch of job "labels" that tell the server only from which source each job came from (Figure 3.14). The goal is, over repeated trials, to converge to a sequencing policy that is optimal with respect to the unknown service-time distributions. Our general approach for solving the scheduling problem considers the multi-armed bandit problem as an abstract model that happens to include the static stochastic scheduling problem (see Figure 3.15).

For example, consider a task model for a case in which each task has "increasing hazard rate." This means that the probability of service completion in the next time step increases monotonically with the amount of service allocated to the task so far.

To test the algorithm, we construct nine task sources with higher task numbers corresponding to higher hazard rates (see Figure 3.16):

$$Pr\{\tau_i = s\} = \{1 - [(1-\rho_i^{(1)})\lambda^{s-1}]\} \prod_{k=1}^{s-1} (1-\rho_i^{(1)})\lambda^{k-1},$$

where $\lambda < 1$; *i.e.,* the probability of task completion increases with the number of task activations.

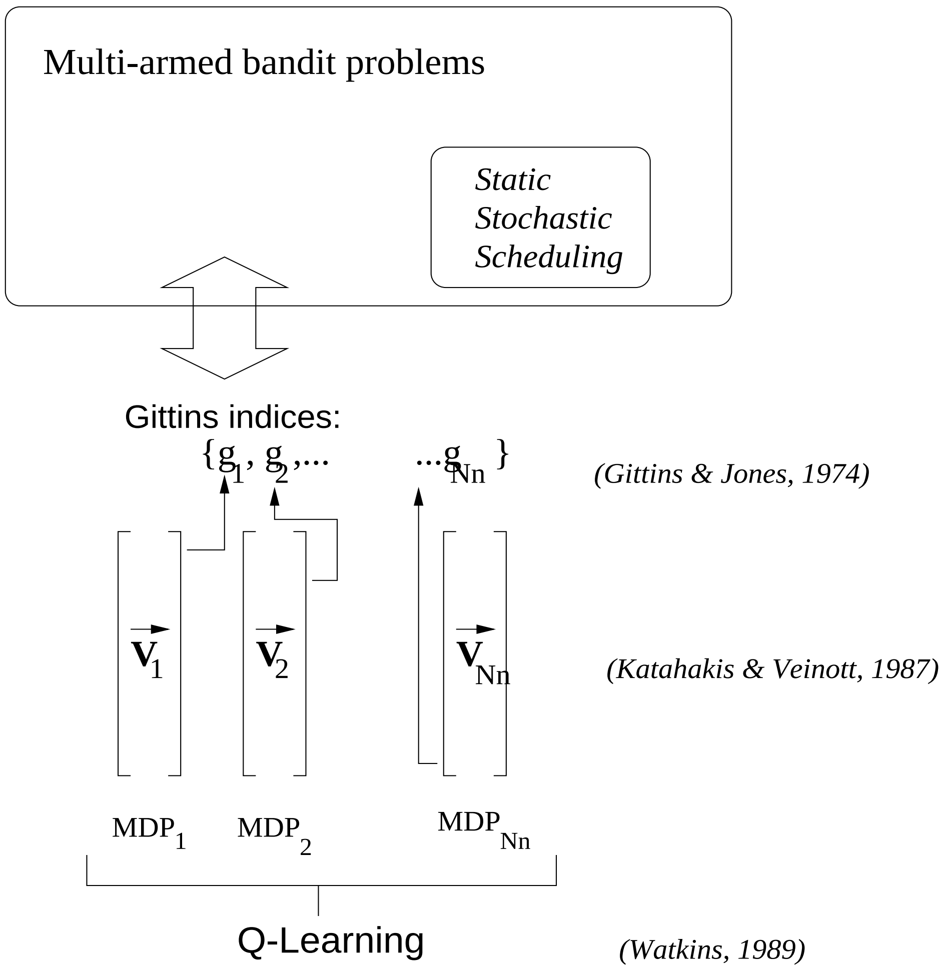


Figure 3.15: Our approach for solving stochastic scheduling problems.

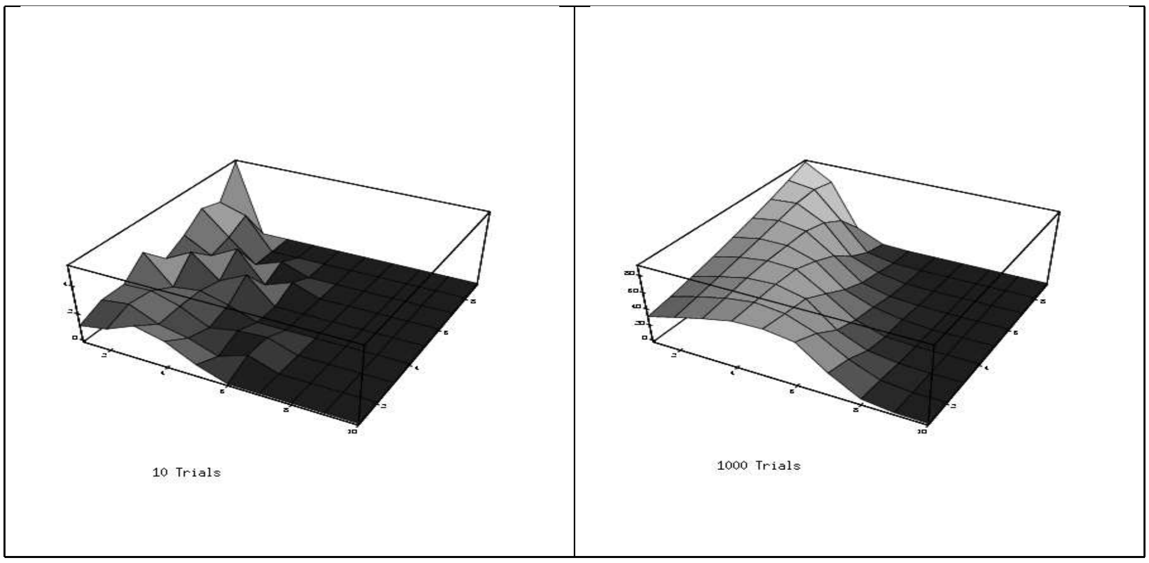


Figure 3.17: Gittins index surface plotted as a function of state (x-axis) and task (y-axis).

The plots of Figure 3.17 appear to be converging to indices that would give rise to such a policy—index estimates for rarely-encountered task-states are slowly rising to their true values. For commonly-encountered bandit states, the Gittins surface estimate yields an optimal scheduling policy relatively early in the learning process. In fact, Figure 3.18 plots the "relative degradation" in performance (the difference in average waiting times under the index policy and the optimal policy, divided by the average waiting time under the optimal policy), and it can be seen that the performance is quite good.

Note that if one were to pursue the straightforward MDP approach mentioned at the end of Section 3.1.1, it would entail a state-set size on the order of fifty million ($\approx 9\times9\times8\times8\times7\times7\times6\times6\times5$) and transition matrices of corresponding dimension.

It is perhaps important to stress that, in the scheduling literature, it is generally assumed that the service-time distributions are known—our reinforcement learning algorithm makes no such assumption. The problem of stochastic scheduling for the cases of constant or monotone hazard-rates is analytically tractable, and the resulting policies are usually somewhat intuitive and can be stated simply. For arbitrary, non-

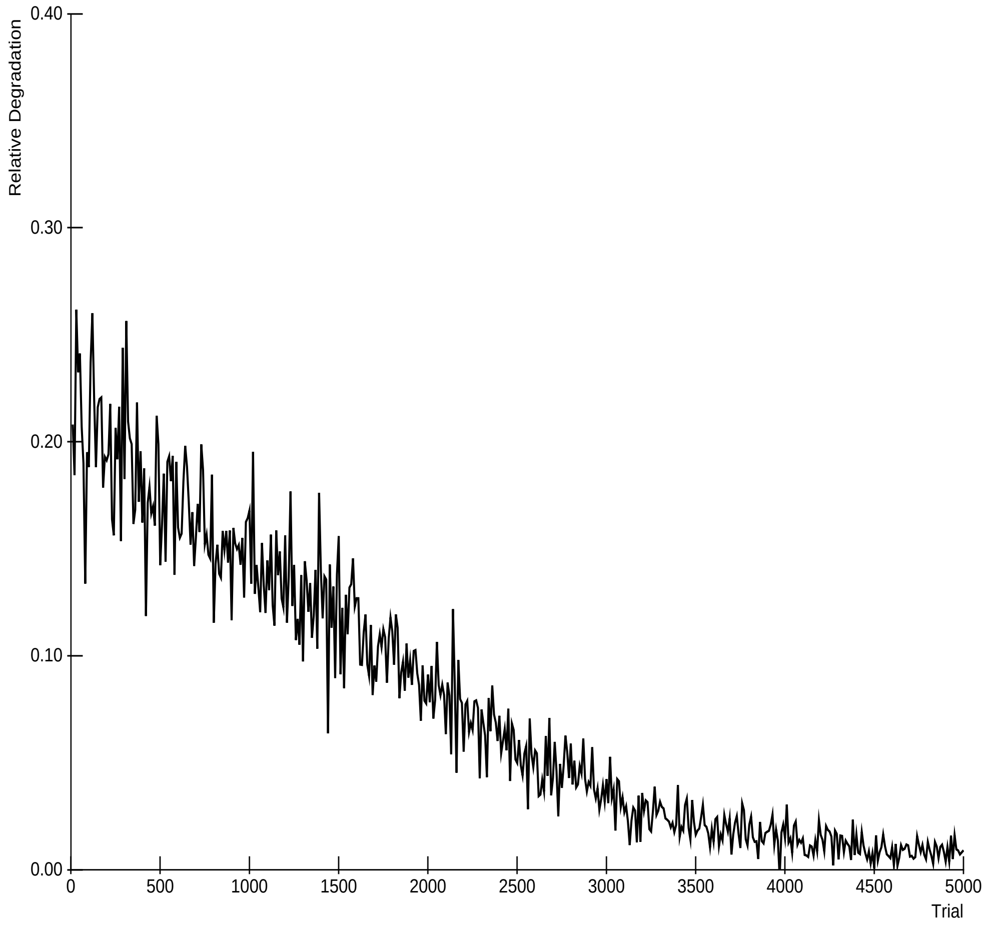


Figure 3.18: Relative degradation versus trial.

monotone hazard-rates, things are less well-understood, but there is nothing in the reinforcement learning approach that would preclude its application to these cases.

The book by Gittins (1989) contains many further applications of the bandit formulation to job scheduling, resource allocation, sequential random sampling, and random search.

### 3.1.6 Q-learning for bandit problems: Summary

Our approach to solving bandit problems has traced the following chain of reasoning:

- A multi-armed bandit is a Markov decision process possessing a special decompositional (Cartesian-product) structure.
- Optimal policies for bandit processes can be constructed efficiently by calculating Gittins indices.
- The Gittins index for a given task state $i$ is also the $i^{th}$ component of the optimal value function for the "restart-in-$i$" problem.
- The restart-in-$i$ process is a standard MDP.
- Optimal policies for MDP's can be computed online in a model-free way using Q-learning.
- Therefore, Q-learning can be applied online, without using a process model, to compute solutions to bandit problems. (The implementational details of a practical algorithm were presented in Section 3.1.4.)

For each alternative $n$-state process, the resulting algorithm computes, in parallel, the desired Gittins indices by solving $n$ two-action MDP's, each of size $n$. A proof of convergence follows, assuming adequate sampling of states and actions, from existing convergence proofs of Q-learning for conventional MDP's (Bertsekas & Tsitsiklis, 1996).

We note that, in practice, the convergence of the our reinforcement learning algorithm may be accelerated through the utilization of parameterized function approximators for representing Q-values. Tsitsiklis and Van Roy (1997) have proposed and analyzed an algorithm that approximates solutions to optimal stopping problems of the sort that underly our approach for computing Gittins indices.[5] Their algorithm utilizes a linear combination of fixed basis functions to approximate the optimal Q-function and an incremental update procedure, similar to conventional Q-learning, for adjusting the weights of the linear combination. Tsitsiklis and Van Roy prove that if Q-function updates are executed for states sampled independently according to the steady-state distribution of the underlying Markov chain, then the weights converge with probability 1. They also provide an error bound for the resulting approximate optimal Q-function. For a Markov chain with $n$ states and a fixed set of $L$ basis functions, $\phi_1(s)$, $\phi_2(s), \ldots, \phi_L(s)$, let $\Phi$ be the $n$-by-$L$ matrix:

$$\Phi = \begin{bmatrix} \phi_1(1) & \cdots & \phi_L(1) \\ \phi_1(2) & \cdots & \phi_L(2) \\ \vdots & \ddots & \vdots \\ \phi_1(n) & \cdots & \phi_L(n) \end{bmatrix}.$$

The vector of Q-value estimates may be written as $\widetilde{Q}(\theta) = \Phi\theta$, where $\theta$ is an $L$-dimensional vector of linear function approximator weights. Let $\|\cdot\|_D$ denote the weighted quadratic norm with respect to the steady state probabilities, $\pi(s)$; that is, $\|x\|_D^2 = \sum_{i=1}^n \pi(i)x_i^2$. Let $\Pi$ denote the operator that projects vectors onto the space of all vectors of the form $\Phi\theta$ with respect to the $\|\cdot\|_D$norm; that is, $\|\Pi x - x\|_D =$

[5]For the stopping problems Tsitsiklis and Van Roy consider, at each stage the agent may STOP and obtain a state-dependent terminal reward, or CONTINUE and make a Markov chain state transition and receive some associated immediate reward. In this case, Q-values under the CONTINUE action may be written as $Q(s, \text{continue}) \equiv Q(s) = \sum_{s'} p_{ss'} \left(r_{ss'} + \gamma \max\{h(s'), Q(s')\}\right)$, where $h(s')$ denotes the terminal reward received if the agent chooses to stop in state $s'$.

$\min_\theta \|\Phi\theta - x\|_D$. If Q-function updates are executed for states sampled independently according to $\pi$, and the columns of $\Phi$ are independent, then the Q-function weights, $\theta$, iteratively generated by Tsitsitklis and Van Roy's Q-learning algorithm converge with probability 1. The limiting vector of weights, $\theta^*$, satisfies the following error bound:

$$\|\Phi\theta^* - Q^*\|_D = \frac{1}{1-\gamma}\,\|\Pi Q^* - Q^*\|_D\,,$$

where $\gamma$ is the discount factor and $Q^*$ is the vector of optimal Q-values. This result is noteworthy in that it establishes a class of sequential decision problems for which an algorithm that combines Q-learning with rather general linear function approximators is guaranteed to converge. In practical terms, the requirement that states be sampled independently from the steady-state distribution may seem overly restrictive. However, we may simply sample states by carrying out a single, infinitely long simulation of the unstopped Markov chain. The sampled states are clearly dependent, but techniques presented by Benveniste, Metivier, and Priouret (1990) can be applied to establish convergence.

We have stressed the model-free nature of our Q-learning approach for bandit problems, but the convergence result for Q-learning with linear function approximators in the context of stopping problems suggests that if we were to incorporate linear function approximators into our Q-learning algorithm for bandit problems, then the resulting scheme may enjoy accelerated convergence over the existing tabular scheme (by virtue of function approximator generalization).

If one has a model for the component bandit processes, real-time dynamic programming (Barto *et al.,* 1991) can be applied, in which full model-based backups are performed for states encountered along sample-paths. Indirect methods, such as adaptive real-time dynamic programming, adaptively construct a model for the controlled process and base control policies and value-function update computations on the latest model (see Barto *et al.*, 1995, for a convergence proof). One advantage of model-free reinforcement learning methods, however, is that, as Monte-Carlo methods, they may inherit some computational advantage over conventional (model-based) methods, particularly for very large problems. This aspect is discussed by Barto and Duff (1994) and Singh (1994).

There are a number of generalizations of the basic bandit formulation that are of extreme practical interest for scheduling. For example, Glazebrook and Gittins (1981) have examined the issue of the existence of index theorems for bandit problems with precedence constraints (their focus is on such constraints that have a tree structure). Whittle (1981) has studied bandit problems in which new tasks arrive (for these "arm acquiring bandits," index results are preserved when the arrival process is Poisson/Bernoulli). The case of context-switching costs has been addressed by Glazebrook (1980). When there is more than one server or processor available—thus enabling more than one process to be active at a time—in general, quite strong additional conditions are required for an index theorem to hold.

An example in Section 3.1.5 considered a problem of stochastic scheduling as a specific instance of the general bandit problem formulation. But general bandit problems themselves are archetypes of "optimal learning" problems, in which the goal is to collect information and use it to inform behavior so as to yield the largest expected reward from actions taken throughout the entire duration of the learning process. We have presented a reinforcement-learning-based algorithm for solving bandit problems, which may be interpreted as an attempt to "learn how to learn optimally." But the reinforcement learning algorithm is itself surely not optimal; its Boltzmann distribution scheme of action-selection is practical and provisional, neither inspired nor informed by a bandit-problem mode of analysis.

One could envision, then, the problem of optimally learning how to learn optimally. (But could one learn how to do this, and do so optimally?...) This regress, as stated, is not entirely meaningful, as Watkins has observed (citing McNamara &

Houston, 1985): "Learning is optimal only with respect to some prior assumptions concerning the... probability distributions over environments the animal [decision-maker] may encounter." In a sense, our underlying Bayesian model of uncertainty "bottoms out" with its implicit assumption of a prior over bandit process parameters. We attempt to compute policies that are optimal with respect to this prior, and while we desire that policy computation be efficient, our policy computation procedure is not intended to be "optimal" (*e.g.*, most efficient in terms of computing time) with respect to any sort of abstracted meta-prior over all possible bandit problems.

## 3.2 Local bandit approximation for optimal learning problems

Watkins (1989) has defined *optimal learning* as: "... the process of collecting and using information during learning in an optimal manner, so that the learner makes the best possible decisions at all stages of learning: learning itself is regarded as a multistage decision process, and learning is optimal if the learner adopts a strategy that will yield the highest possible return from actions over the whole course of learning."

We presented a number of examples of optimal learning problems in Chapter 1, ranging from simple bandit problems to more complex Bayes-adaptive Markov decision processes. In this chapter, we have seen that, for the special class of bandit problems, efficient techniques exist for computing optimal policies. For problems with physical state, however, such as Markov decision processes with uncertain transition probabilities, computation grows increasingly burdensome with problem size, and so one is compelled to seek approximate solutions, some of which may ignore the effects of information gain entirely. In contrast, the approach we shall propose in the remaining part of this chapter explicitly acknowledges that there is an information-gain component to the optimal learning problem. If certain salient aspects of the value of information can be captured, even approximately, then one may be led to a reasonable method for approximating optimal learning policies.

For multi-armed bandit problems, the information pattern is the sole component of the hyperstate, and this special subclass of MDP's has tractably-computable optimal learning strategies. Our approach in this section attempts to make use of this fact. Actions for general BAMDP's are derived by, first, attaching to a given BAMDP in a given state a "local" n-armed bandit process that captures some aspect of the value of information gain as well as explicit reward. Indices for the local bandit model can be computed relatively efficiently; the largest index suggests the best action in an optimal-learning sense. The resulting algorithm has a receding-horizon flavor in that a new local-bandit process is constructed after each transition; it makes use of a mean-process model as in some previously-suggested approximation schemes, but here the value of information gain is explicitly taken into account, in part, through index calculations.

The most obvious difference between the optimal learning problem for a BAMDP and the multi-armed bandit problem is that the BAMDP has a physical, Markov-chain state component to its hyperstate. A first step in bandit-based approximation, then, proceeds by "removing" this physical-state component. This can be achieved by viewing the process on a time-scale defined by the recurrence time of a given state. For example, suppose the process is in some state, $s$. In response to some given action, two things can happen (Figure 3.19):

1. The process can transition, in one time-step, into $s$ again with some immediate reward, or

2. The process can transition into some state, $s' \neq s$, and experience some "sojourn" path of states and rewards before returning to $s$.

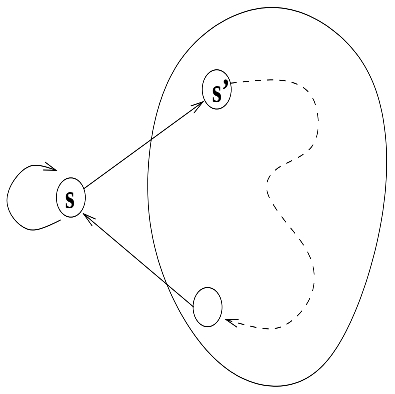
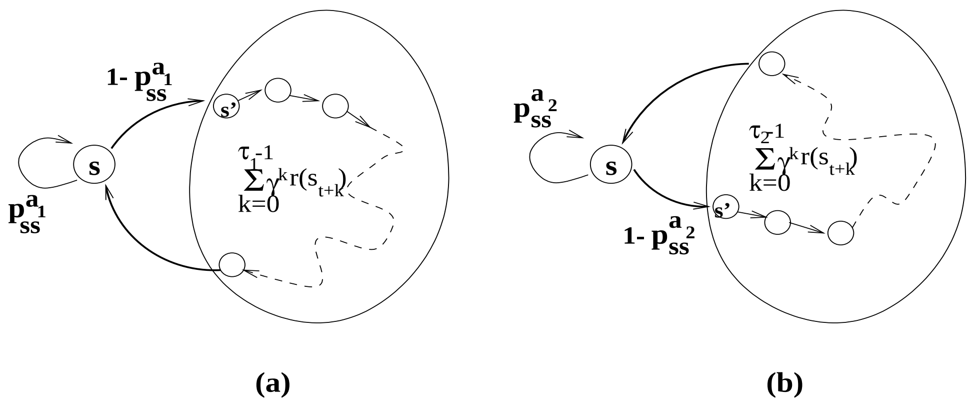


Figure 3.19: Conversion from multi-state Markov to single-state semi-Markov.



$1-p_{ss}^{a_1}$
$p_{ss}^{a_1}$
s
s'
$\sum_{k=0}^{\tau_1-1}\gamma^k r(s_{t+k})$
(a)
$p_{ss}^{a_2}$
$1-p_{ss}^{a_2}$
s
s'
$\sum_{k=0}^{\tau_2-1}\gamma^k r(s_{t+k})$
(b)

Figure 3.20: Semi-Markov bandit processes associated with taking (a) ACTION 1, and (b) ACTION 2.

On a time-scale defined by sojourn-time, one can view the process in a sort of "state-$s$-centric" way. From this perspective, the process appears to have only one state, and is semi-Markov; that is, the time between transitions is a random variable. Some other action taken in state $s$ would give rise to a different sojourn reward process (Figure 3.20). For both processes, the sojourn path/reward will depend upon the policy for sojourn states, but suppose that this policy is fixed for the moment. By viewing the original process on a time-scale of sojourn-time, one has effectively removed, or "collapsed," the physical-state component of hyperstate. The new process has one state, $s$, and the problem of choosing an action, given that one is uncertain about the transition probabilities, presents itself as a semi-Markov bandit problem.

The preceeding discussion suggests a local-bandit-model-based algorithm for approximating optimal learning policies for BAMDP's:

**(0)** Given that uncertainty in transition probabilities is expressed in terms of sufficient statistics, $M$, and the process is currently in state $s_t$.

**(1)** Compute the optimal policy for the certainty-equivalent process, $\bar{P}(M)$, in which unknown transition probabilities are set to their Bayesian point estimates:

$$\pi^*[\bar{P}(M)]$$

(this defines a nominal policy for sojourn states).

**(2)** Construct a local bandit model at state $s$; that is, the decision maker must choose between some number (the number of admissible actions) of sojourn reward processes—this is a semi-Markov multi-armed bandit problem.

**(3)** Compute the Gittins indices for the local bandit model.

**(4)** Take the action with the largest index.

**(5)** Observe a transition to $s_{t+1}$ in the underlying MDP.

**(6)** Update $M$ accordingly (Bayes update).

**(7)** Go to step (1)

For example, consider the simple BAMDP depicted in Figure 3.21, and suppose the process is currently in the left-hand state, STATE 1. The local semi-Markov bandit process associated with STATE 1 / ACTION 1 MDP is shown in Figure 3.22. The sufficient statistics for $p_{11}^1$ are denoted by $(\alpha, \beta)$, and $\frac{\alpha}{\alpha+\beta}$ and $\frac{\beta}{\alpha+\beta}$ are the expected probabilities for transition into STATE 1 and STATE 2, respectively. $\Gamma$ and $R_{121}$ are random variables signifying sojourn time and reward (computed with respect to maximum-likelihood estimates for $P$). The true infinite-horizon bandit process

With probability $\frac{\alpha}{\alpha+\beta}$ the underlying Markov chain makes a self-transition, and the associated information-state component, $(\alpha, \beta)$ becomes $(\alpha+1, \beta) \neq s^*$. With probability $\frac{\beta}{\alpha+\beta}$, the underlying Markov chain transitions to STATE 2 and eventually back to STATE 1, at which point the information-state component of hyperstate changes to $(\alpha, \beta+1) = s^*$. The underlying chain evolves under the influence of transition probabilities defined by this new information-state until a sojourn path is completed, signifying a transition to a hyperstate $\neq s^*$. The new reward rate for node $(\alpha, \beta)$ can be computed by referring to Figure 3.24(b) ($\hat{p}$-quantities signify sojourn transition probabilities defined with respect to the certainty-equivalent optimal policy). The numerator of the expression for reward rate is just the discounted value for being in STATE 1, while the denominator is the value of STATE 1 if rewards in the chain depicted in 3.24(b) are all set to 1.

## 3.3 Summary

This chapter has considered multi-armed bandit problems, an important subclass of Bayes-adaptive Markov decision processes whose optimal policies may be defined in terms of Gittins indices. We have presented a reinforcement-learning-based algorithm for computing these indices that makes use of a particular characterization of indices as solutions to a collection of low-dimensional stopping problems. The algorithm performs parallel Q-value updates utilizing bandit process sample-path data.

Our algorithm is model-free; it does not require the specification of bandit process state transition probabilities (though, if these probabilities were specified, our algorithm could incorporate full model-based backups). The example of static stochastic scheduling with unknown service-time distributions provides a meaningful instance of a problem in which the model-free aspect of our algorithm plays a necessary role. We note that the application of our algorithm to the classical Bernoulli bandit problem would not require the specification of a prior, rather, over the course of learning, policies greedy with respect to evolving Q-values would converge to optimal strategies consistent with the distribution of bandit processes presented to the learner and actually experienced; *i.e.*, "Nature's" prior.

In the last section, we suggested how bandit process models and Gittins indices might be brought to bear upon the intractable problem of computing optimal policies for general BAMDP's. By viewing BAMDP's on a time-scale of recurrence or sojourn time with respect to the current physical state, we essentially collapse all other physical states, and thus remove the physical-state component of the hyperstate. The result is a local *semi*-Markov bandit process model, whose Gittins indices (and policies based upon them) acknowledge some aspect of information gain.

One might hope for the existence of a generalized Gittins index result that holds for processes with physical state, but unfortunately, none are known to exist. Statistical independence of alternative processes appears to be a necessary requirement for the existence of optimal index rules. Also, it is known that a geometrically-discounted reward criterion is necessary for the existence of Gittins indices (Barry & Fristedt, 1985).

# CHAPTER 4

# REINFORCEMENT LEARNING ALGORITHMS FOR BAMDP'S

In this chapter, we pursue a somewhat direct approach for computing approximately-optimal policies for Bayes-adaptive Markov decision processes. We propose two algorithms; both appeal directly to the full-blown, BAMDP version of Bellman's equation defined over hyperstates, but our algorithms use parameterized function-approximators and Monte-Carlo sampling to sidestep Bellman's "menace of the expanding grid."

To help motivate our first algorithm, we begin by considering stochastic policies for MDP's. We develop a policy-iteration algorithm for improving these policies, and derive a sample-based version of the algorithm. We then adopt parameterized function approximators for representing policies and value functions, integrate these approximators into the stochastic policy iteration scheme, employ a Monte-Carlo approach for estimating the value function gradient with respect to policy parameters, and apply the resulting "actor-critic, policy-gradient" algorithm to BAMDP's.

Our second algorithm is a relatively straightforward application of SARSA($\lambda$) (which was discussed in Section 1.2.7) to BAMDP's. For each physical-state and action, we adopt a separate linearly-parameterized function approximator for generalizing evolving Q-value estimates over their information state components.

## 4.1 Tabular Stochastic Policy Iteration: Offline and online algorithms based upon gradient projection and ascent

In this section we first develop an offline, model-based algorithm for stochastic policy iteration, applying Rosen's (1960) feasible-directions gradient projection method of nonlinear programming to the special case of stochastic-policy Markov decision processes with known transition probabilities. Adopting a relatively informal perspective, we then translate our algorithm to an online, sample-based (model-free) scheme.

### 4.1.1 Introduction

Consider the case of a discounted Markov decision process. There are three classical approaches for computing an optimal policy: (1) Value iteration (Blackwell, 1965), in which the method of successive approximation is applied to Bellman's equation, (2) Policy iteration (Howard, 1960), which may be regarded as a function-space version of Newton's method for solving Bellman's equation (Puterman & Brumelle, 1979), and (3) Linear programming (d'Epenoux, 1960), in which a positive linear combination of value function components is minimized subject to a set of linear constraints (which express the fact that if $V$ is an upper bound for $r + \gamma PV$ for all policies, then $V$ is an upper bound for the optimal value function).

For an MDP with $N$ states and $M$ possible actions, value iteration requires $O(MN^2)$ computation per iteration. Convergence of the value function to its optimal value is asymptotic; the rate of convergence is governed by the subdominant eigenvalue of the transition matrix associated with the optimal policy.

Policy iteration alternates a greedy policy improvement step (whose computation is similar to that of a single iteration of value iteration) with evaluation of the improved policy, which requires the solution of a linear system of equations, an $O(N^3)$ computation. Since an improved policy is generated in every evaluation/improvement cycle, and the number of policies is finite, policy iteration is guaranteed to converge and terminate in a finite number of iterations.

The performance of these two algorithms can be improved in a variety of ways; for example, in value iteration suboptimal actions can be eliminated using information gathered in the natural course of computation (MacQueen, 1966). An $O(N^2)$ iterative method can compute an approximate solution to the linear system that policy iteration must solve (Puterman & Shin, 1978), or state aggregation can reduce the dimension of the linear system (Bertsekas & Castanon, 1989).

Linear programming is considered to be impractical for large problems, though its theory facilitates sensitivity analysis and the incorporation of constraints. The dual formulation constraint matrix has $N$ rows and $NM$ columns. The simplex method applied to a problem of this size requires $O(N^3)$ computation on average but is exponential in the worst case, while Karmarkar's (1984) rescaling algorithm leads to an $O(N^3)$ worst-case bound.

It is known that the maximum of the expected total discounted reward over the set of deterministic stationary policies equals that over the set of all policies. Value iteration and policy iteration consider only "pure" deterministic policies as feasible solutions. Basic feasible solutions of the dual linear programming problem correspond to stationary deterministic policies—the simplex method considers only pure policies as it proceeds. General feasible solutions to the dual LP can be identified with randomized stationary policies (via "state-action" frequencies; see, for example, Derman and Stauch, 1966). Therefore, Kamarkar's method, which operates in the interior of the feasible set, could be said to be the only version of a conventional method for discounted MDP's that considers randomized policies.

The approach pursued in this section explicitly takes the space of feasible policies to be that of stationary randomized policies. Beginning with a feasible stochastic policy, an offline model-based algorithm computes the associated value function and the gradient of the value function with respect to policy parameters. Gradient components are then projected onto the space of active constraints and the resulting direction is adopted as a feasible direction of ascent. It will be argued that the resulting algorithm requires $O(N^3)$ computation, which is comparable to conventional policy iteration, and empirical evidence may lead one to conjecture that, with some mild caveats, it converges to a globally-optimal policy. Later in this section, we shall provide a translation of the offline algorithm to a sample-based (model-free), online version.

Motivation for this investigation stems from philosophically-related efforts to make use of "smooth" models for BAMDP's (see Chapter 6). In contrast to their deterministic counterparts, stochastic policies are specified by parameters that exist in a topological space endowed with a metric and may be viewed as varying in a smooth, continuous manner with its arguments. These aspects allow computational techniques based directly upon smooth, continuous analytic methods to be applied.

### 4.1.2 Stochastic policies

In general, the indices $i$ and $j$ shall refer to Markov chain states and range from 1 to $N$. The index $k$ shall refer to possible actions, and range from 1 to $M$. A distinguished, fixed state will be denoted as $s$ or $s'$, while a distinguished, fixed action will be denoted as $a$. State-related indices will appear as subscripts, while action-related indices will appear as superscripts.

We shall be considering strategies defined by stationary stochastic policies, which may be specified by $NM$ numbers ($N(M-1)$ values suffice):

$$\pi_i^k = Pr\{action = k | state = i\} \quad i = 1, ..., N \quad k = 1, ..., M,$$

which must satisfy the constraints of being a collection of valid probability mass functions; *i.e.*,

$$\begin{aligned} \textstyle\sum_k \pi_i^k &= 1 \quad i = 1, ...N \\ \pi_i^k &\geq 0 \quad i = 1, .., N \quad k = 1, ..., M. \end{aligned}$$

One reason we do this is because we shall be utilizing gradient ascent to improve the policies, and the gradient of value with respect to an abstract action (e.g., ACTION= "TURN_LEFT") is not well-defined, while the gradient with respect to the *probability* of taking an abstract action *is* well-defined.

#### 4.1.2.1 Value

For a fixed stationary stochastic policy, the expected infinite-horizon sum of discounted rewards, or value function, $V_i,\ i = 1, ..., N$, satisfies a linear system of equations expressing local consistency:

$$V_i = \sum_k \pi_i^k \left[ \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j) \right] \quad i = 1, ..., N,$$

which may be rewritten explicitly as a linear system of the form $Ax = b$, with

$$\begin{aligned} A_{ij} &= \delta_{ij} - \gamma \textstyle\sum_k \pi_i^k p_{ij}^k \\ b_i &= \textstyle\sum_k \pi_i^k \left( \sum_j p_{ij}^k r_{ij}^k \right). \end{aligned}$$

The matrix $A$ may be interpreted as the "resolvent" matrix, $I - \gamma P_\pi$, associated with the stochastic policy $\pi$, while $b$ is the vector of expected immediate rewards under $\pi$.

#### 4.1.2.2 The gradient

We now derive an expression for components of the gradient of the value function with respect to the action probabilities, $\frac{\partial V_i}{\partial \pi_s^a}$. Taking partial derivatives of both sides of the value function equation leads to:

$$\begin{aligned} \frac{\partial V_i}{\partial \pi_s^a} &= \gamma \textstyle\sum_j \sum_k \pi_i^k p_{ij}^k \frac{\partial V_j}{\partial \pi_s^a} & i \neq s \\ \frac{\partial V_s}{\partial \pi_s^a} &= \gamma \textstyle\sum_j \sum_k \pi_s^k p_{sj}^k \frac{\partial V_j}{\partial \pi_s^a} + \sum_j p_{sj}^a \left( r_{sj}^a + \gamma V_j \right) & i = s \end{aligned}$$

which may be cast explicitly as a linear system of the form $Ax = b$, with

$$\begin{aligned} A_{ij} &= \delta_{ij} - \gamma \textstyle\sum_k \pi_i^k p_{ij}^k \\ b_i &= \delta_{is} \left[\textstyle\sum_j p_{sj}^a \left(r_{sj}^a + \gamma V_j\right)\right]. \end{aligned}$$

There is one such $N$-dimensional system for each $(s, a)$ pair, and there are $MN$ such pairs. Note, however, that the matrix $A$ does not depend upon $s$ or $a$, and so its inverse need only be computed once. Only the vector $b$ changes with each $(s, a)$ pair, and only one component of $b$ is nonzero—its value is the Q-value $Q_V(s, a)$. This implies that the gradient components are simply columns of $A^{-1}$ (the resolvent of the policy-averaged transition matrix) scaled by the Q-values; that is,

$$\begin{aligned} \tfrac{\partial V_s}{\partial \pi_{s'}^a} &= \left[\textstyle\sum_j p_{s'j}^a (r_{s'j}^a + \gamma V_j)\right] \times \left[A^{-1}\right]_{ss'} \\ &= Q_V(s', a) \times \left[A^{-1}\right]_{ss'} \end{aligned}$$

and thus the computational complexity of computing the gradient is $O(N^3)$.

#### 4.1.2.3 Policy improvement

In the policy improvement step for conventional (deterministic) policy iteration, the policy is improved by selecting the action, $a$, in each state, $s$, that maximizes $r(s, a) + \gamma \sum_j p_{sj}^a \widetilde{V}_j$, where $\widetilde{V}$ denotes the value function associated with the current unimproved policy. There is a rigorous sense in which this step can be identified with a step of a generalized version of Newton's method (Puterman & Brumelle, 1978), but more informally, the policy improves by moving greedily in a "direction" towards $\arg\max_a \, r(s, a) + \gamma \sum_j p_{sj}^a \widetilde{V}_j$, or even more informally, in the direction "$\frac{\partial}{\partial a}\left[r(s, a) + \gamma \sum_j p_{sj}^a \widetilde{V}_j\right]$"—quotes signify that the partial derivative is not well defined, since the action set is not a metric space.

Turning now to the case of stochastic policies, for each state $s$, we step greedily in the direction suggested by $\frac{\partial V_s}{\partial \pi_s^k}$, $k = 1, ..., M$. Note that this direction considers only components of the gradient that are associated with action probabilities in state $s$, though components $\frac{\partial V_s}{\partial \pi^k_{s'}}$, $s' \neq s$, are routinely nonzero.

### 4.1.3 Gradient projection

Naively moving the policy in the direction $\frac{\partial V_s}{\partial \pi^k_s}$, $k = 1, ..., M$ will in general result in violation of the constraints that $\{\pi^a_s\}^N_{k=1}$ specifies a valid probability distribution. To maintain feasibility, we project the gradient onto a subspace defined by an appropriate set of active constraints. Essentially, this is an application of Rosen's (1960) gradient-projection algorithm applied to the special case of stochastic policy MDP's.

Rosen's method is a nonlinear programming algorithm for solving problems of the form:

$$\begin{array}{llll} maximize & f(x) & & \\ subject & Ax & \leq & b \\ & Cx & = & e. \end{array}$$

By "active" constraints we shall mean the equality constraints together with binding inequality constraints. Initially, the algorithm seeks a feasible direction, $d$, with $\nabla^\top f(x)d > 0$, where $d$ lies in the tangent subspace generated by active constraints—the algorithm starts by determining a feasible $d$ so that movement in its direction will result in an increase in $f$ and all active constraints remain active.

Let $\widetilde{A}$ be defined as a matrix constructed from the rows of active constraints; *i.e.*, $C$ augmented by rows of $A$ for which the inequality constraints hold with equality. It can be shown that the projection of the gradient onto the tangent subspace defined by active constraints is:

$$\begin{aligned} d &= \left[I - \widetilde{A}^\top(\widetilde{A}\widetilde{A}^\top)^{-1}\widetilde{A}\right]\nabla f \\ &= P\nabla f, \end{aligned}$$

where $P$ is the *projection matrix* corresponding to the tangent subspace defined by $\widetilde{A}$. It can also be shown that if $d = P\nabla f \neq 0$, then $d$ provides a feasible direction of ascent. If $d = 0$, then it may be possible to relax one of the active inequality constraints and move in a new direction to an improved point— this can be accomplished by examining components of $\beta = (\widetilde{A}\widetilde{A}^\top)^{-1}\widetilde{A}\nabla f$ associated with active inequality constraints ($\beta$ is computed as a side-effect of computing $P$) . If all such components are positive then the Kuhn-Tucker conditions are satisfied and the process terminates. If one or more of these components are negative, then we repeatedly delete the row of $\widetilde{A}$ corresponding to the most negative component (we relax the constraint) and project $d$ onto the remaining subspace. More details of the algorithm are supplied by standard nonlinear programming texts (*e.g.*, Luenberger, 1973).

The most demanding step of the projection procedure is the computation of $(\widetilde{A}\widetilde{A}^\top)^{-1}$; however, if this quantity is known for a given $\widetilde{A}$, and if $\widetilde{A}$ is then augmented by a row drawn from the inequality constraint matrix (or if a row of $\widetilde{A}$ is excised), then the new value of $(\widetilde{A}\widetilde{A}^\top)^{-1}$ can be calculated by a simple update procedure. For the case of our stochastic policy MDP, the equality constraint matrix $A$ is simply a row-vector of 1's, while the inequality matrix $C = -I$ . These special characteristics further simplify the projection calculation:

- Initially, $(\widetilde{A}\widetilde{A}^\top)^{-1} = \frac{1}{m}$ if none of the inequality constraints are binding for the initial policy; for example, if the initial action distribution is uniformly distributed.

- A careful analysis reveals that if we let $A_1 = \widetilde{A}$ (with $q$ rows), and let $A_2 = A_1$ augmented by a newly "activated" row drawn from the inequality constraint matrix, then

$$(A_2A_2^\top)^{-1} = \begin{bmatrix} \xi & \xi y^\top \\ \xi y & Z + yy^\top \end{bmatrix},$$

where

$$\begin{aligned} Z &= (A_1 A_1^\top)^{-1} \\ y &= \begin{bmatrix} z_{11} \\ \vdots \\ z_{q1} \end{bmatrix} \\ \xi &= \tfrac{1}{1-z_{11}}. \end{aligned}$$

(this part of the projection matrix does not depend upon the exact identities of the active inequality constraints, only their number; the full-set of $M$ matrices, $(\widetilde{A}\widetilde{A}^\top)^{-1}$, which are symmetric and of size $q \times q$ for $q = 1, 2, \ldots, M$, could be pre-calculated).

### 4.1.4 Summary of offline stochastic policy iteration

- Initialize the stochastic policy to be uniform over all actions.
- Repeatedly,
  - Compute the value function associated with the current stochastic policy by solving a system of linear equations, $Ax = b$, with

    $$\begin{aligned} A_{ij} &\equiv \delta_{ij} - \gamma \textstyle\sum_k \pi_i^k p_{ij}^k \\ b_i &\equiv \textstyle\sum_k \pi_i^k \left( \sum_j p_{ij}^k r_{ij}^k \right). \end{aligned}$$

  - Compute the gradient components, $\frac{\partial V_s}{\partial \pi_s^a}$, $s = 1, ..., N$, $a = 1, ..., M$ using our previous formula evaluated for "diagonal" components:

    $$\begin{aligned} \tfrac{\partial V_s}{\partial \pi_s^a} &= \left[ \textstyle\sum_j p_{sj} (r_{sj}^a + V_j) \right] \times \left[ A^{-1} \right]_{ss} \\ &= Q_V(s, a) \times \left[ A^{-1} \right]_{ss}. \end{aligned}$$

  - For each $s = 1, ..., N$, project the gradient $\frac{\partial V_s}{\partial \pi_s^k}$, $k = 1, ..., M$ onto a tangent subspace of active constraints (this may involve the application of an existing projection matrix, or if the set of binding inequality constraints has changed, it may require a projection matrix update).

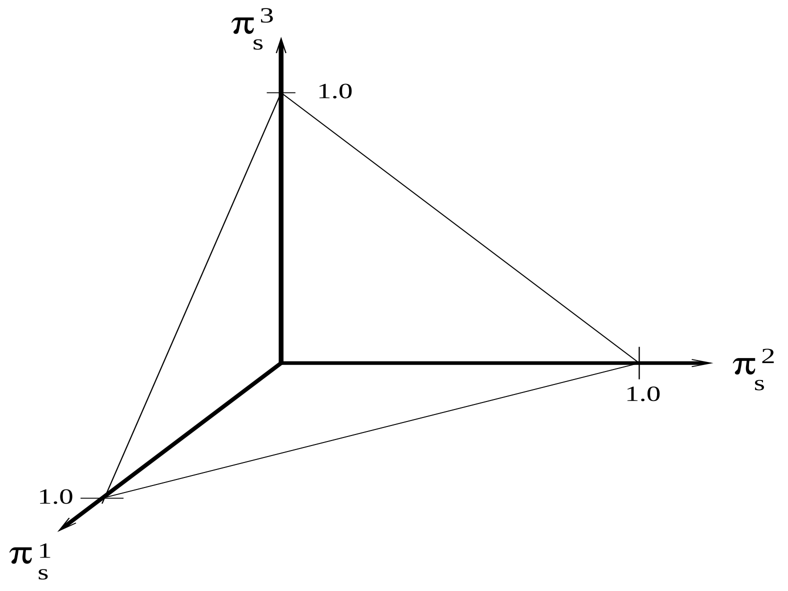


Figure 4.1: Stationary stochastic policies are specified by state-action probability parameters drawn from the feasible set (the diagonal facet).

- Perform a line-search in the resulting direction, $d$, (or determine a policy-step size in some other way, see the discussion below), and step along $d$ to an improved value.

### 4.1.5 Example

We randomly generate an MDP with four states and three actions in each state. For each state, the feasible set of action parameters is the diagonal facet of a triangularly-faced simplex in three-dimensions (see Figure 4.1).

We plot the action probabilities in barycentric coordinates (which correspond to locations in the feasible action set) as the stochastic policy iteration algorithm progresses—the action probabilities for all states are plotted, with distinct labels, in the same coordinate system.

One detail that we have neglected to address in detail is that of step size; *i.e.*, how far should the algorithm step in the direction $d$? Primal nonlinear programming methods like Rosen's typically employ a line search routine that, given a feasible step-length interval and direction $d$, returns the step length along $d$ that maximizes

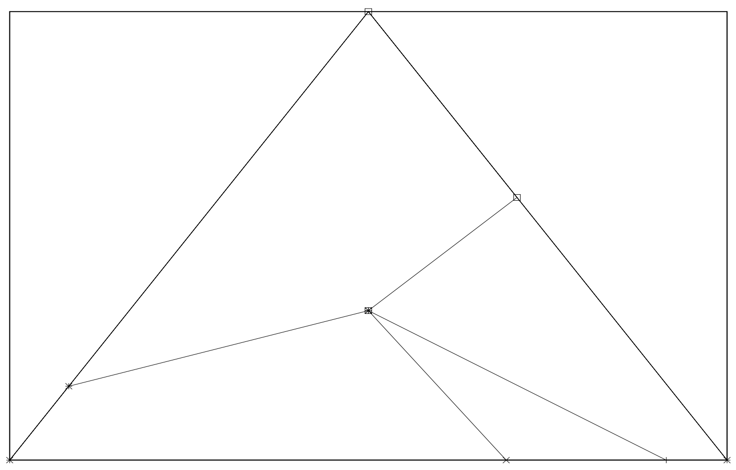


Figure 4.2: Stochastic policy iteration with line search to determine step length.

the objective function. Figure 4.2 shows the result of doing this for our example. The policy moves immediately to a constraint boundary, and on the second step follows the boundary to the optimal solution. This behavior differs from what conventional (deterministic) policy iteration would do. The algorithm starts from the interior of the feasible set, and the first step leads, in this example, to a policy in which, for each state, one of the feasible actions has been eliminated (it has zero probability).

The projected gradient does not appear to change much (until a new constraint becomes active), at least in this example. If we replace the line search with a fixed small step-size, then the general shape of the policy trajectory differs only slightly from the line-search case (Figure 4.3). This suggests that line-search may be unnecessary. We could simply compute the step size required to reach the nearest inequality constraint boundary along direction $d$.

### 4.1.6 Online sample-based translation

The stochastic policy iteration algorithm described so far is interesting in its own right, and further experiments could be conducted to gain a deeper understanding of its behavior, but we now move on to construct an online, sample-based (model-free) version.

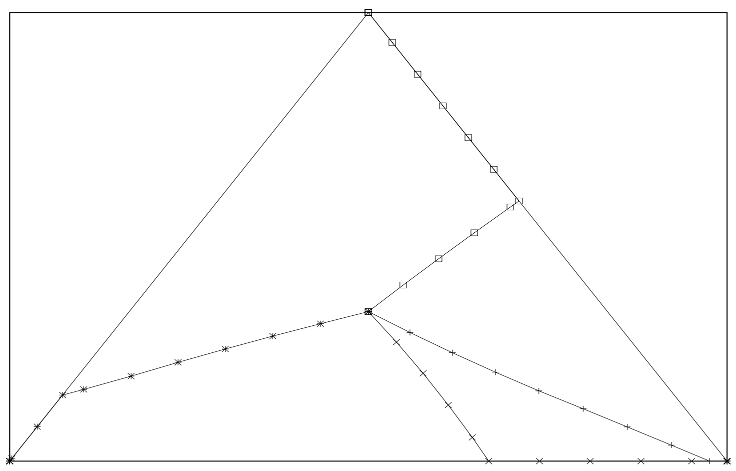


Figure 4.3: Stochastic policy iteration with fixed small step-size = .1.

#### 4.1.6.1 TD calculation of the gradient

We begin by reviewing our perspective on the method of temporal differences that was presented in Section 1.2.2. Consider a Markov chain augmented by a reward structure (one step expected rewards are denoted $r_{ij}$). The expected total discounted reward, $V$, satisfies a system of linear equations:

$$V_i = \sum_j p_{ij}(r_{ij} + \gamma V_j) \quad i = 1, ..., N.$$

Since the right-hand side of this system, viewed as an operator, is a contraction, successive approximation is one viable method for computing a solution:

$$V_i^{(n+1)} = \sum_j p_{ij}(r_{ij} + \gamma V_j^{(n)}) \quad i = 1, ..., N.$$

Now suppose we observe a single transition of the chain, $s \to s'$. Based upon this single transition, our instantaneous view of the transition matrix is that $p_{ss'} = 1$, but is otherwise zero. If this were truly the case, then successive approximation would suggest an update of the form:

$$V_s \leftarrow r_{ss'} + \gamma V_{s'}.$$

In practice, we must severely under-relax the update so as to average out the sampling noise. The result is (with $\alpha$ some small scalar):

$$\begin{aligned} V_s &\leftarrow (1-\alpha)V_s + \alpha(r_{ss'} + \gamma V_{s'}) \\ &\leftarrow V_s + \alpha(r_{ss'} + \gamma V_s - V_{s'}), \end{aligned}$$

which is the method of temporal-differences (Sutton, 1988), TD($\lambda$), with $\lambda = 0$.

Now reconsider the stochastic policy iteration algorithm. In each iteration, two systems of linear equations need to be solved: one for the value function and one for the gradient of that value function. One can now apply our TD-derivation rationale to derive sample-based TD-like rules for both systems.

For the value function system, if we apply action $a$ and observe an $s \to s'$ transition, the implied corresponding TD backup is:

$$V_s \leftarrow (1-\alpha)V_s + \alpha\left[\pi_s^a(r_{ss'}^a + \gamma V_{s'})\right].$$

For the gradient system, we update $mn$ components:

$$\begin{array}{lll} \frac{\partial V_s}{\partial \pi_i^k} \leftarrow (1-\alpha)\frac{\partial V_s}{\partial \pi_i^k} + \alpha\gamma\pi_s^a\frac{\partial V_{s'}}{\partial \pi_i^k} & & i \neq s \ \ or \ \ k \neq a \\ \frac{\partial V_s}{\partial \pi_s^a} \leftarrow (1-\alpha)\frac{\partial V_s}{\partial \pi_s^a} + \alpha\left[\gamma\pi_s^a\frac{\partial V_{s'}}{\partial \pi_s^a} + (r_{ss'}^a + \gamma V_{s'})\right] & & i = s \ \ and \ \ k = a. \end{array}$$

(Alternatively, taking $\pi_s^a \equiv 1$ in both the value and the gradient backup equations, a measure that is also consistent with our informal perspective, seems to make little difference in the resulting policy-parameter trajectories.)

#### 4.1.6.2 Algorithm summary

- Initialize the stochastic policy to be uniform over all actions.
- Repeatedly,
  - Choose an action, $a$, via the current stochastic policy,

– Observe a state transition $s \to s'$ and reward $r^a_{ss'}$,

– Perform a value function TD backup:

$$V_s \leftarrow (1-\alpha)V_s + \alpha\left[\pi^a_s(r^a_{ss'} + \gamma V_{s'})\right],$$

– Perform $MN$ gradient component TD backups:

$$\begin{array}{lcll} \frac{\partial V_s}{\partial \pi^k_i} & \leftarrow & (1-\alpha)\frac{\partial V_s}{\partial \pi^k_i} + \alpha\gamma\pi^a_s\frac{\partial V_{s'}}{\partial \pi^k_i} & i \neq s \ \ or \ \ k \neq a \\ \frac{\partial V_s}{\partial \pi^a_s} & \leftarrow & (1-\alpha)\frac{\partial V_s}{\partial \pi^a_s} + \alpha\left[\gamma\pi^a_s\frac{\partial V_{s'}}{\partial \pi^a_s} + (r^a_{ss'} + \gamma V_{s'})\right] & i = s \ \ and \ \ k = a, \end{array}$$

– Project the gradient components $\frac{\partial V_s}{\partial \pi^k_s}$ $k = 1, ..., M$, as in Section 4.1.4, to determine a feasible direction of ascent $d$,

– Step the policy components, $\pi^k_s$, $k = 1, \ldots, M$, in direction $d$.

#### 4.1.6.3 Example

Figure 4.4 plots the performance of the online sample-based version of stochastic policy iteration applied to our four-state / three-action example. The relaxation parameter, $\alpha$, and policy step size were set somewhat arbitrarily (value and gradient systems have the same $\alpha = .1$), and are surely not optimal; nevertheless, the algorithm converged to the optimal policy after only eighty online state transitions. With four states and three actions per state, this works out to an average of less than seven samples per state-action.

### 4.1.7 Stochastic policy iteration: Summary

The basic offline, model-based stochastic policy iteration algorithm presented first in this section may be viewed as a somewhat straightforward assault upon the problem of optimizing Markov decision processes, based upon an application of Rosen's gradient-projection method of nonlinear programming. We derived a simple expression for the gradient of the value function with respect to state-action probabilities, and an efficient method for computing gradient components. The special form of the

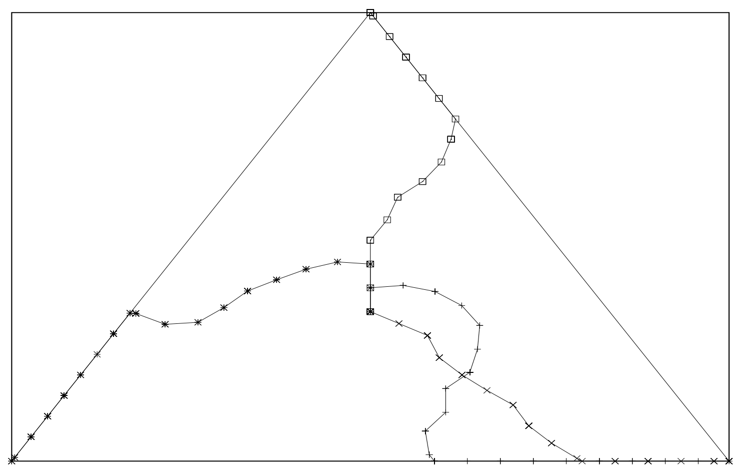


Figure 4.4: Performance of online sample-based stochastic policy iteration.

constraint set associated with feasible stochastic policies simplifies the computation of the projection matrix, which projects the value function gradient onto the relevant tangent subspace. Adopting a relatively informal or intuitive perspective (which one suspects could be justified rigorously by appealing to the theory of stochastic approximation), we then passed from the offline algorithm to an online, sample-based (model-free) version.

The convergence properties of Rosen's algorithm have, apparently, yet to be fully resolved. Theoretically, the algorithm is not "closed" (since the method for selecting an ascent direction changes abruptly when the set of active constraints changes) and is therefore subject to the possibility of "jamming."[1]

The stochastic policy MDP considered here is a special case of a more general problem treated by Rosen, and the implications of this, together with the fact that our algorithm is strongly motivated by deterministic policy iteration, which is globally convergent, may lead one to conjecture that, given reasonable assumptions concerning policy step-size, relaxation parameters, and initial policy, our stochastic pol-

[1]"It remains an open question as to whether the gradient-projection method can experience jamming, for although no convergence proof is available, neither is there a counterexample. Indeed, the method has been used successfully for numerous nonlinear programming problems and jamming has never been detected."—(Luenberger, 1973, p. 251).

icy iteration algorithms converge to globally-optimal policies. It has been argued that the offline stochastic policy iteration algorithm requires $O(N^3)$ computation, just as for deterministic policy iteration, and its convergence behavior for our example seems comparable to the deterministic version, though the example may be somewhat misleading (the three-action case was chosen because policies can be plotted in two-dimensions). In general, the algorithms proceed in quite different ways. The deterministic version visits only the corners of the feasible action parameter simplex, while the stochastic version may move through the interior through a series of intermediate parameter values lying on the facets or edges as well.

We have shown how the computation of the projection matrix simplifies for the case of our stochastic policy Markov decision process—it was shown how we can revise $(\widetilde{A}\widetilde{A}^\top)^{-1}$ using a simple and efficient update when a new constraint becomes active. Further improvements in the projection procedure can probably be developed. It is unclear whether, for the case of stochastic policy Markov decision processes, gradient projection ever produces an ascent step in which a previously-active inequality constraint is relaxed and becomes inactive. If such a scenario is impossible, then the projection algorithm becomes even less complex.

Unlike the offline algorithm, the current version of the online algorithm maintains estimates of all $MN^2$ gradient components, though only $MN$ of them (and only $M$ of these at each transition step) are explicitly used to determine an ascent direction. Clearly, it would be desirable if we could reduce the computation required by $MN$ gradient estimate backups at each transition step.

## 4.2 A policy-gradient algorithm for Bayes-adaptive Markov decision processes

We now derive a simulation-based stochastic policy iteration algorithm suitable for approximating optimal policies for Bayes-adaptive Markov decision processes. In this approach, we approximate value functions and policies by functions involving linear combinations of information state components, and we employ a Monte-Carlo technique for estimating components of the value function gradient with respect to policy parameters.

### 4.2.1 Value and value-gradient with respect to policy parameters

We first present a natural extension of our tabular stochastic policy iteration algorithm to a version that incorporates parameterized function approximation for representing value functions and policies. Expressions for value and gradient of value with respect to policy parameters differ only slightly from the tabular case.

#### 4.2.1.1 Stochastic policy and value

We recall that a stochastic policy may be specified by: $\pi_i^k = Pr\{action = k|state = i\}$, $i = 1, ..., |S|$, $k = 1, ..., |A|$, where $|S|$ and $|A|$ are the number of states and actions respectively. The value function, $V_i^\pi$, $i = 1, ..., |S|$, is the expected infinite-horizon discounted sum of rewards, starting in state $i$ and applying a fixed policy $\pi$. Value function components satisfy a linear system of equations that enforces local consistency:

$$V_i^\pi = \sum_k \pi_i^k \left[ \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j^\pi) \right] \quad i = 1, ..., |S|.$$

In the context of BAMDP's, the state set, $S$, is the collection of hyperstates, which even for simple problems, is quite large. Suppose we represent policies by

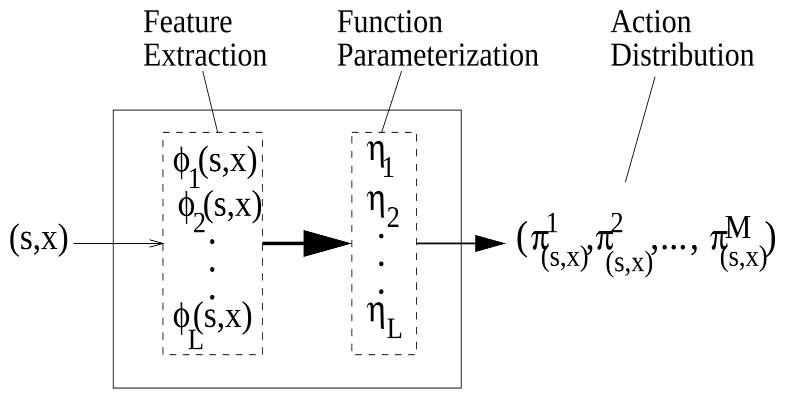


Figure 4.5: A general architecture for policy parameterization. Given a hyperstate as input, the function approximator outputs a probability distribution over admissible actions. We employ a similar architecture for representing value functions (the output in this case is a real-valued approximation of the hyperstate's value).

functions that depend upon parameters $\eta_1, \eta_2, ..., \eta_L$, where $L < |S| (|A| - 1)$ :

$$\pi_i^k(\eta_1, ..., \eta_L) \qquad L << |S| .$$

One may think of the $\eta$'s as parameters that, together with a collection of features, determine distributions over actions given a hyperstate as input (Figure 4.5).

With this reduced degree-of-freedom, we cannot specify all possible stochastic policies exactly (in particular, in general we cannot specify the optimal policy), but our hope is that our parameterized family of policies will be rich enough to represent policies that are good or nearly optimal. In exchange, our policy representation, which is now biased, becomes independent of the number of states.

Consider the state-action probabilities to be implicit functions of the parameter vector $\eta$; that is, $\pi_i^k = \pi_i^k(\eta_1, \eta_2, ..., \eta_L)$, $i = 1, ..., |S|$, $k = 1, ..., |A|$. Taking partial derivatives of both sides of the value-function equation with respect to a generic component of the policy parameter vector leads to:

$$\begin{aligned} \frac{\partial V_i^\pi}{\partial \eta_l} &= \sum_k \frac{\partial \pi_i^k}{\partial \eta_l} \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j^\pi) + \sum_k \pi_i^k \frac{\partial}{\partial \eta_l} \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j^\pi) \\ &= \sum_k \frac{\partial \pi_i^k}{\partial \eta_l} \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j^\pi) + \gamma \sum_j \sum_k \pi_i^k p_{ij}^k \frac{\partial V_j^\pi}{\partial \eta_l}. \end{aligned} \tag{4.1}$$

This describes an $|S| \times |S|$ linear system of equations for the components $\frac{\partial V_i^\pi}{\partial \eta_l}$, $i = 1, 2, ..., |S|$. It has the form $Ax = b$, where

$$\begin{aligned} A_{ij} &= \delta_{ij} - \gamma \sum_k \pi_i^k p_{ij}^k \\ b_i &= \sum_k \frac{\partial \pi_i^k}{\partial \eta_l} \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j). \end{aligned} \tag{4.2}$$

There are $L$ such linear systems, one corresponding to each $\eta_l$.

#### 4.2.1.2 Monte-Carlo gradient computation

$A$ is a square matrix with dimension size equal to the number of hyperstates. Solving for the vector of gradient components directly (an $O(|S|^3)$ proposition) is unthinkable in general (since $|S|$ is the number of hyperstates) as are iterative methods, which require $O(|S|^2)$ operations per iteration. However, a Monte-Carlo approach leads to a tractable computational procedure.

Note that the matrix $A$ is of the form

$$A = I - \gamma P_\pi,$$

where $P_\pi$ is the "policy-averaged" transition matrix; *i.e.*, it is the average transition matrix associated with the stochastic policy $\pi$. Let us re-express the solution of the linear system in terms of the Neumann series:

$$\begin{aligned} Ax &= b \\ x &= A^{-1} b \\ &= (I - \gamma P_\pi)^{-1} b \\ &= \sum_{k=0}^{\infty} (\gamma P_\pi)^k b. \end{aligned}$$

Our goal is to maximize the value associated with the initial hyperstate, $i_0$. Therefore, we are interested in determining the gradient of this value component with respect to controller parameters. Notice that $\frac{\partial V_{i_0}}{\partial \eta_l}$ is the $i_0$th element of $A^{-1}b$, or equivalently, it is the ($i_0th$ row of $A^{-1}$) $\times b$. A Monte-Carlo scheme can make use of this fact. For example, for an episodic, undiscounted ($\gamma = 1$) process, the Neumann series expression above implies that the $(i_0, j)th$ component of $A^{-1}$ is just the expected number of visits to hyperstate $j$, given that the process starts in $i_0$.[2] This implies that we may obtain an unbiased sample of $\frac{\partial V_{i_0}}{\partial \eta_l}$ by simulating a hyperstate trajectory starting in $i_0$ and accumulating components of the $b-$vector corresponding to the trajectory's hyperstates:

$$\sum_{j \;\; on \;\; trajectory} b_j.$$

Note that the $A$ matrix is the same for all gradient components. Only the $b$-vector depends upon the parameter, $\eta_l$. Therefore a single trajectory yields unbiased estimates for *all* gradient components, $\frac{\partial V_{i_0}}{\partial \eta_l}$ $\forall l$.

Conceptually, Equation 4.1 defines $L$ linear systems of equations; each linear system has the form $Ax = b$, where the solution, $x$, is the vector of gradient components, $\frac{\partial V_i}{\partial \eta_l}$, $i = 1, \ldots, N$ for fixed choice of $l$. Each of the $L$ linear systems has the same $A$-matrix, but the $b$-vectors vary with $l$ as specified by Equation 4.2. In our Monte-Carlo approach, we simulate a hyperstate trajectory starting from $i_0$, which determines the element indices of $b$ that are to be accumulated to form an unbiased estimate of $\frac{\partial V_{i_0}}{\partial \eta_l}$ $\forall l$. This method for selecting element indices of $b$ is valid for *all* of the $b$-vectors corresponding to different choices of $\eta_l$, and so, operationally, a single hyperstate trajectory is sufficient for providing an unbiased estimate of gradient components with respect to all $\eta_l$.

[2] To convert discounted value problems to undiscounted problems (1) Define an auxiliary terminal state, with associated reward of zero, (2) Scale all transition probabilities by the discount factor, $\gamma$, and (3) Define new transition probabilities to the terminal state (from all previously-existing states) to be $1-\gamma$. Operationally, we can terminate the simulated trajectory with probability $1-\gamma$.

### 4.2.2 A policy-gradient algorithm for Bayes-adaptive Markov decision processes

We next apply our function-approximation version of simulation-based stochastic policy iteration to the problem of approximating optimal policies for BAMDP's. A simple example illustrates how our algorithm can overcome computational barriers encountered by a classical dynamic programming approach.

#### 4.2.2.1 Bayes-adaptive Markov decision processes

In Chapter 1, we defined a Bayes-adaptive Markov decision process (BAMDP) as a Markov decision processes with Bayesian modeling of unknown transition probabilities. The status of such a process at a given stage is summarized by its *hyperstate*, which consists of the process's Markov chain state, or *physical state*, together with its *information state*, which is the collection of distributions describing uncertainty in the transition probabilities. Given the hyperstate, if we apply an action and observe a transition, then Bayes's rule prescribes how the information state is to be revised—it is its prior multiplied by the likelihood function associated with the observed transition, normalized. In practice, it is useful to adopt a conjugate family of distributions to represent the information state. The revision of a member of a conjugate family of distributions in light of observation requires only a simple update of defining parameters. We then view the information state as simply the collection of all parameter values that define the distributions describing our uncertainty.

Figure 4.6 depicts the transition diagram for a simple MDP with two physical states, two feasible actions in each state, and transition probabilities that are unknown. Our goal is to maximize the expected total (undiscounted) reward received over the entire course of interaction, which we assume has finite duration. Observing the process is equivalent to observing four different Bernoulli processes: STATE 1 / ACTION 1, STATE 2 / ACTION 1, STATE 1 / ACTION 2, and STATE 2 / ACTION 2. The beta distribution forms a conjugate family of distributions with respect to

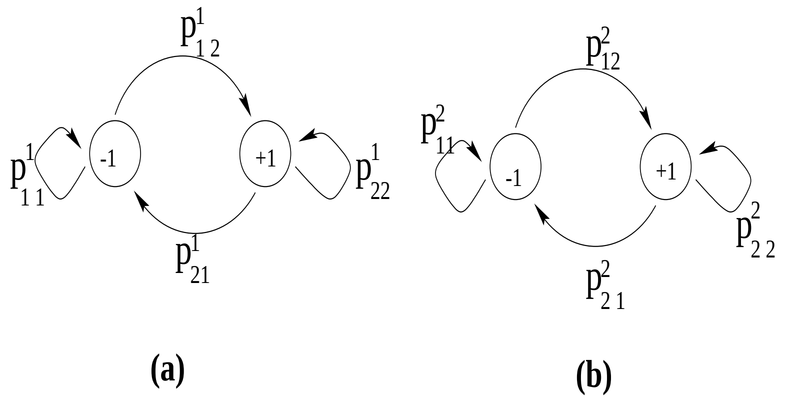


Figure 4.6: An MDP with uncertain transition probabilities: dynamics under (a) action 1 and (b) action 2 ($\pm 1$ denotes the rewards for entering right- and left-hand states).

Bernoulli sampling. We denote the information state by a pair of $2 \times 2$ matrices, whose entries parameterize the beta distributions describing our uncertainty—the parameters are directly related to observed transition counts under action 1 and action 2 respectively.

The associated BAMDP transition diagram is depicted in Figure 4.7. The transition diagram is acyclic, and the number of hyperstates grows exponentially with the time-horizon (Bellman's "menace of the expanding grid").

We now integrate parameterized value and gradient estimates developed in Section 4.2.4 with the stochastic policy iteration algorithm presented in Section 4.1.5, and apply the resulting algorithm to this Bayes-adaptive Markov decision processes.

#### 4.2.2.2 Function approximation issues

We shall employ distinct parameterized function approximators for representing value functions and stochastic policies for the BAMDP. First, with regard to value-function approximation, we propose that our approximator be a linear combination of hyperstate features; *i.e.*, $V(s,x) \approx \sum_l \theta_l \varphi_l(s,x)$. We choose this architecture for its simplicity, and useful analytical convergence results exist for this case.

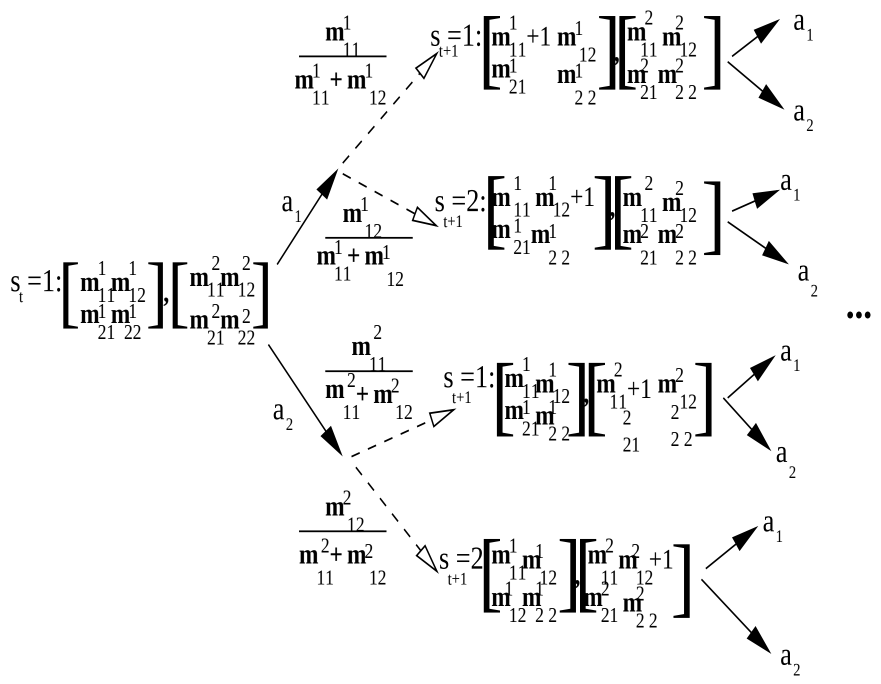


Figure 4.7: Transition diagram for a simple BAMDP.

We proceed under the assumption that value is a relatively smooth function of information state.[3] Without further (problem-specific) assumptions, little can be said in general about behavior as a function of physical state. Therefore, we propose using separate parameterized value-function approximators for each physical state. For our feature set, we propose using the components of information state, $x$, together with a "bias" feature grounded at a value of 1—the total number of feature-vector components is thus $N^2M + 1$, where $N$ is the number of physical states. We then write:

$$V(s, x) \approx V_s(x) = \sum_l \theta_l[s] x_l,$$

where $V_s$ denotes the function-approximator associated with physical state $s$, and $x_0 = 1$ while $x_l$ for $l > 0$ ranges over information state components.

[3]It is known that the value function, $V$, is a continuous function of information state parameters, $x$ (Martin, 1967, p.44). See also Figure 1.12 in Chapter 1.

Turning now to the issue of function approximation for policies, we propose that stochastic policy functions, which map hyperstates and collections of admissible actions to probability distributions over admissible actions, be relatively smooth functions of information state, but allow for arbitrary and discontinuous variation with respect to physical state and prospective action. We therefore propose using a separate parameterized function approximator for each physical state and each admissible action. Since we have committed to using parameterized function approximation for representing policies, we choose an architecture that produces only valid distribution functions over candidate actions. In anticipation of utilizing the gradient formula developed in Section 4.2.1, we also require that our approximator be differentiable with respect to its parameters. Many policy approximation architectures satisfy these criteria—here we consider one possibility, an exponentiated, normalized function involving exponential terms that are linear in the feature components:

$$\pi\left((s,x),a\right) \approx \pi_{(s,a)}\left(x\right) = \frac{e^{\sum_l \eta_l[s][a]x_l}}{\sum_{a'} e^{\sum_l \eta_l[s][a']x_l}},$$

where $\pi_{(s,a)}$ signifies the function approximator associated with state $s$ and action $a$, and where $l$ ranges over the $N^2M+1$ components of information state and bias.

In Section 4.1, we adopted a tabular representation for representing stochastic policies for MDP's, which required the explicit storage of action probabilities for each state. In the tabular case, policy-improvement increments suggested by analytic gradient formulae must be projected in such a way that the policy remains stochastic (a viable probability distribution over admissible actions). The resulting gradient-projection policy improvement scheme appears to be globally convergent, but for BAMDP's a tabular representation is not feasible, since the number of hyperstates is, in general, so large. By adopting the exponentiated, normalized functional representation for policies, memory requirements are drastically reduced, and we no longer require a gradient projection step to maintain the policy's probability distribution property, but the basic act of employing (biased) parameterized function approximators implies that we can no longer claim that the resulting algorithm is globally convergent (Section 4.4 includes further discussion of convergence issues).

#### 4.2.2.3 Policy and value gradients

We now apply our stochastic policy iteration algorithm to BAMDP's, where policies are represented as above. The main analytical step consists of calculating the various components $\frac{\partial \pi_i^k}{\partial \eta_l}$ appearing in Section 4.2.1's formula for value-gradient. Applying the quotient rule, considering the numerator of the result for possible cases of $i$ and $k$, and using our definition of $\pi$ results in:

$$\begin{aligned}
\frac{\partial}{\partial \eta_j[i][k]} \pi_{(s,a)}(x) &= \frac{\partial}{\partial \eta_j[i][k]} \frac{e^{\sum_l \eta_l[s][a] x_l}}{\sum_{a'} e^{\sum_l \eta_l[s][a] x_l}} \\
&= \begin{cases} \pi_{(s,a)}(x) \left[\delta_{ka} - \pi_{(s,k)}(x)\right] x_j & i = s \\ 0 & i \neq s \end{cases}
\end{aligned}$$

where $\delta$ is the Kronecker delta.

Recall that the Monte-Carlo scheme estimates gradient components via

$$\frac{\partial V_{i_0}}{\partial \eta_l} \approx \sum_{i \in traj} b_i,$$

where

$$b_i = \sum_k \frac{\partial \pi_i^k}{\partial \eta_l} \sum_j p_{ij}^k (r_{ij}^k + \gamma V_j).$$

Previously, we provided a Monte-Carlo interpretation of our value-gradient formula. With hyperstates $(s, x)$ sampled by following policy $\pi$ from the initial state, $(s(t_0), x(t_0))$, the incremental contribution to the gradient estimate (reflecting one hyperstate transition, $(s, x) \to (s', x')$) becomes:

$$\Delta \frac{\partial}{\partial \eta} V^\pi\left(s(t_0), x(t_0)\right) \propto \sum_a \frac{\partial \pi_{(s,a)}(x)}{\partial \eta} \sum_{s'} p_{ss'}^a(x) \left[r_{ss'}^a + V^\pi\left(s', x'(s, a, s', x)\right)\right].$$

Here, $p_{ss'}^a(x)$ denotes the Bayesian point estimate for a physical state transition from $s$ to $s'$ under action $a$ given information state $x$, and $x'(s,a,s',x)$ denotes the Bayes-rule-updated information state induced by a physical-state transition in the underlying Markov chain from $s$ to $s'$ under action $a$, given the prior information-state, $x$. Substituting our expression for the policy gradient, $\frac{\partial \pi}{\partial \eta}$, leads to:

$$\begin{aligned}
\Delta \frac{\partial}{\partial \eta_j[s][[k]} V^{\pi}\left(s(t_0), x(t_0)\right) \;\; &\propto \;\; \textstyle\sum_a \{ \; \pi_{(s,a)}\left(x\right) \left[\delta_{ka} - \pi_{(s,k)}\left(x\right)\right] \\
&\qquad \times \textstyle\sum_{s'} p^a_{ss'}(x) \left[r^a_{ss'} + V^{\pi}\left(s', x'(s,a,s',x)\right)\right] \} \, x_j \\
&= \;\; \pi_{(s,k)}\left(x\right) \{ \textstyle\sum_{s'} p^k_{ss'}(x) \left[r^k_{ss'} + V^{\pi}\left(s', x'(s,k,s',x)\right)\right] \\
&\quad - \textstyle\sum_a \pi_{(s,a)}\left(x\right) \sum_{s'} p^a_{ss'}(x) \left[r^a_{ss'} + V^{\pi}\left(s', x'(s,a,s',x)\right)\right] \} \, x_j.
\end{aligned}$$

#### 4.2.2.4 Algorithm summary

Our Monte-Carlo algorithm for approximating optimal policies for BAMDP's is sample-based with separate parameterized function approximators for value and for policy. It proceeds by simulating trajectories of the BAMDP: starting from the the initial hyperstate—the physical state $s(t_0)$ and information state $x(t_0)$, which defines the prior—it follows the hyperstate trajectory under policy $\pi$. At each step of the trajectory,

- Let the current hyperstate be $(s,x)$. Suppose the value function parameters are $\theta_l[s]$, $s = 1.2, ..., N$, $l = 0, 1, 2, ..., L$, and the policy parameters are $\eta_l[s][a]$, $s = 1, 2, ..., N$, $a = 1, 2, ..., M$, $l = 0, 1, 2, ..., L$, where $L = N^2 M$ is the number of information state components.

- Generate action $a$ via random sampling from the Boltzmann-type distribution:

$$\pi_{(s,a)}\left(x\right) = \frac{e^{\sum_l \eta_l[s][a] x_l}}{\sum_{a'} e^{\sum_l \eta_l[s][a'] x_l}},$$

- Observe $(s,x) \rightarrow (s',x')$ and reward $r$.

- Update value function parameters (via a TD update):

$$\theta_j[s] \leftarrow \theta_j[s] + \alpha \left( r + \sum_l \theta_l[s']x'_l - \sum_l \theta_l[s]x_l \right) x_j \qquad j = 0, 1, 2, ...L,$$

  where $\alpha$ is some small step size.

- Update policy parameters; for $k = 1, 2, ...M$ and $j = 0, 1, 2, ..., N^2M$:

$$\begin{aligned} \eta_j[s][k] \quad \leftarrow \quad & \eta_j[s][k] + \beta\pi_{(s,k)}(x) \left\{ \textstyle\sum_{s'} p^k_{ss'}(x) \left( r^k_{ss'} + \sum_l \theta_l[s']x'_l(s,k,s',x) \right) \right. \\ & \left. - \textstyle\sum_a \pi_{(s,a)}(x) \sum_{s'} p^a_{ss'}(x) \left( r^a_{ss'} + \sum_l \theta_l[s']x'_l(s,a,s',x) \right) \right\} x_j \end{aligned}$$

  where $\pi$ is specified by our Boltzmann-type approximator, and $\beta$ is some small step-size (smaller than $\alpha$ makes sense and works well).

#### 4.2.2.5 Example

We now apply our algorithm to the HORIZON=25 case of the example presented in Figures 4.6 and 4.7. We choose to accumulate the parameter changes suggested by our algorithm over the course of each simulated trajectory, and then scale those changes by total change-magnitude before making a true parameter change; *i.e.*, we perform a “batch” update of the parameters. This is consistent with calling $\alpha$ and $\beta$ “step-sizes”—results are plotted in Figure 4.8.

Each point plotted represents an estimate of expected total reward under the evolving policy—each point is the sample average, over 100,000 Monte-Carlo trials, of total reward obtained for a given policy. We note that there is non-negligible variance, or jitter, of this sample-mean performance, even though we are averaging over a relatively large number of trials. Part of this jitter is due to changes in the policy as it improves with increasing trial number (points are plotted every tenth trial). But we also note that the variance of the sample mean of total reward, $Var(\overline{R})$, is proportional to the variance of total reward; *i.e.*,

$$Var(\overline{R}) = \frac{Var(R)}{N},$$

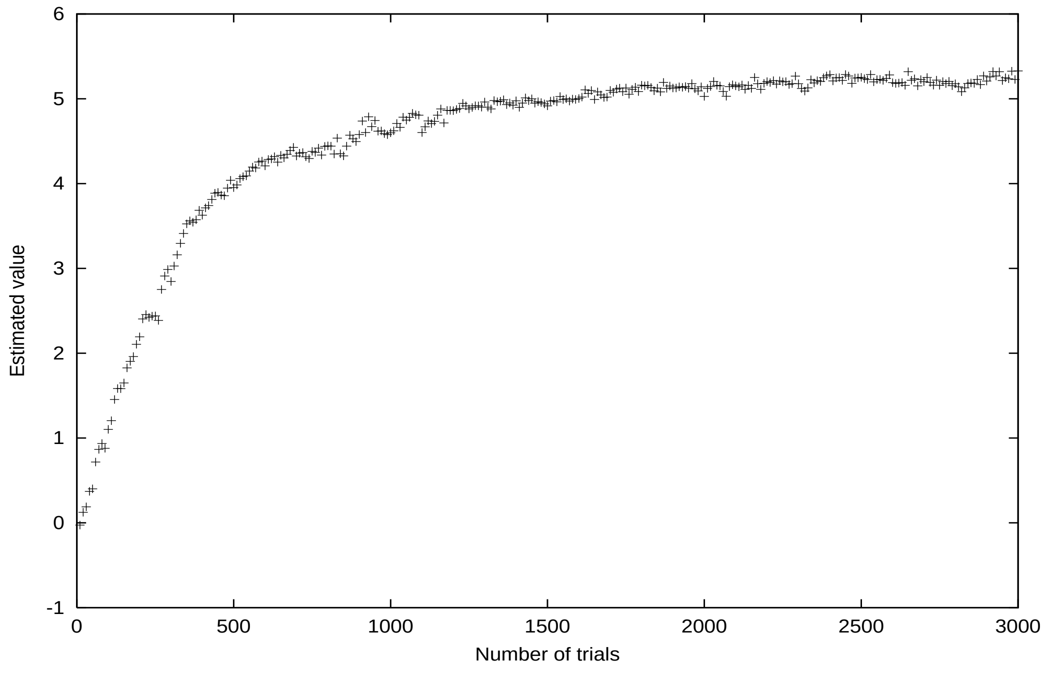


Figure 4.8: Performance of simulation-based stochastic policy iteration ($\alpha = .05$, $\beta = .01$) on the 2-state, 2-action example, with uniform priors for unknown transition probabilities. For this 2-state/2-action horizon=25 problem, there are nearly 2 million distinct reachable hyperstates. Value and policy function approximators for this case used only 18 and 36 parameters, respectively, but achieved performance that is within 4% of optimal.

where $R$ is the total reward and $N$ is the number of Monte-Carlo trials utilized in computing $\overline{R}$.

How large is $Var(R)$? That is, how large is the variance of total reward for the BAMDP governed by a fixed policy?

Suppose we consider the policy derived after 3000 trials. The sample variance of total reward for the BAMDP governed by this policy is approximately 117. Hence, in utilizing $100,000$ Monte-Carlo trials to estimate expected total reward, we would expect a deviation of about $\sqrt{\frac{117}{100,000}} \cong .034$ in the plot of sample-mean value.

Performance is not quite that of the optimal tabular policy (which has value 5.719 and was computed by a single backward Gauss-Seidel sweep over the 2 million hyperstates). Our compressed policy representation does not have sufficient degrees-of-freedom to reproduce the optimal policy, which can tailor actions to every specific hyperstate, but its performance is within approximately 4% of optimal.

## 4.3 SARSA($\lambda$) for Bayes-adaptive Markov decision processes

Recall from Section 1.2.5 that SARSA is an "on-policy" version of Q-learning; that is, in the context of MDP's, we maintain estimates of Q-values,

$$Q_V(s,a) = \sum_{s'} p^a_{ss'} \left( r^a_{ss'} + \gamma V(s') \right),$$

which gauge the value of taking action $a$ in state $s$, and then executing a policy with value $V(\cdot)$ thereafter. If the process in is state $s$ and we take action $a$, and the process transitions into state $s'$ and yields reward $r$, the SARSA Q-value update is

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma Q(s', a') - Q(s,a) \right],$$

where $a'$ is the action actually taken at the next step from state $s'$.

For BAMDP's, the number of hyperstates is typically very large, and so, as with our previous policy-gradient approach, we shall adopt a linear function approximator for representing (Q-) value functions. Again, we proceed under the assumption that Q-values are relatively smooth functions of information state, $x$, but that variation with respect to the physical state component of hyperstate, or with respect to action, may be arbitrary. We therefore propose using separate parameterized Q-value function approximators for each physical state and each action:

$$Q\left((s,x),a\right) \approx Q_{(s,a)}(x) = \sum_l \eta_l[s][a]x_l.$$

As before, our feature set is composed of the components of information state, $x$, together with a "bias" feature grounded at a value of 1. There are thus $N^2M+1$ feature-vector components, with corresponding function approximator parameters, $\eta_l[s][a]$, $l=0,1,\ldots,L$, $s=1,2,\ldots,N$, $a=1,2,\ldots,M$.

Utilizing this linear approximation architecture changes SARSA's update from an update of Q-values (stored in a table) to an update of the Q-value parameters vector, $\overrightarrow{\eta}$. Observing a hyperstate transition, $(s,x)\ {}^a\!\rightarrow_r\ (s',x')$, gives rise to a parameter update of the form:

$$\overrightarrow{\eta}[s][a] \leftarrow \overrightarrow{\eta}[s][a] + \alpha\left[r+\gamma Q_{(s',a')}(x') - Q_{(s,a)}(x)\right]\nabla_{\overrightarrow{\eta}[s][a]}Q_{(s,a)}(x).$$

This is SARSA with linear-function approximators in the context of BAMDP's.

In practice, we may derive improved performance of the algorithm by maintaining decaying traces of the gradient vector, $\nabla_\eta Q$. At the beginning of each trial, we set all traces, $e_l[s][a]$, $l=0,1,\ldots,L$, $s=1,2,\ldots,N$, $a=1,2,\ldots,M$, to zero. After each step, in which we apply action $a$ in hyperstate $(s,x)$ and observe $(s,x)\ {}^a\!\rightarrow_r\ (s',x')$, we decay all traces by a factor of $\gamma\lambda$, reset the trace associated with state $s$, action $a$,

$$\overrightarrow{e}[s][a] = \nabla_{\overrightarrow{\eta}[s][a]}Q_{(s,a)}(x),$$

and clear other traces associated with actions not taken in state $s$ :

$$\vec{e}[s][\bar{a}] = 0 \qquad \forall \bar{a} \neq a.$$

Motivation for resetting the traces in this way comes from empirical evidence obtained for the tabular (non-function approximation) version of the algorithm (see Sutton & Barto, 1998, Section 7.8 and pp. 212-213).

### 4.3.1 Algorithm summary

- Initialize the Q-value function parameter vector, $\vec{\eta}[s][a]$, randomly, and set all traces, $\vec{e}[s][a]$, to zero.
- For each trial (simulation of hyperstate trajectory to terminal horizon):
  - Let $(s, x)$ be the initial hyperstate, $(s(t_0), x(t_0))$.
  - Compute the Q-value estimates for each action:

    $$Q_{(s,a)}(x) = \sum_l \eta_l[s][a]x_l \qquad \forall a.$$

  - Compute the $\epsilon$-greedy action; *i.e.*, $a = \arg\max_a Q_{(s,a)}(x)$; with probability $\epsilon$, choose $a$ randomly.
  - For each step of the trial:
    * Decay all traces, $\vec{e} = \gamma\lambda\vec{e}$.
    * Reset traces associated with state $s$, action $a$ :

      $$\begin{aligned} \vec{e}[s][a] &= \nabla_{\vec{\eta}[s][a]} Q_{(s,a)}(x) \\ \vec{e}[s][\bar{a}] &= 0 \qquad \forall \bar{a} \neq a. \end{aligned}$$

    * Apply action $a$, and observe hyperstate transition $(s, x) \ {}^{a} \rightarrow_r \ (s', x')$.

* Compute the Q-value estimates for each action, $a'$. in the new hyperstate, $(s', x')$:

$$Q_{(s',a')}(x') = \sum_l \eta_l[s'][a']x'_l \qquad \forall a'.$$

* Compute the $\epsilon$-greedy action; *i.e.*, $a' = \arg\max_{a'} Q_{(s',a')}(x')$; with probability $\epsilon$, choose $a'$ randomly.

* Update the Q-value function approximator parameters:

  For $s = 1, 2, \ldots, N$, $a = 1, 2, \ldots, M$,

$$\vec{\eta}[s][a] \leftarrow \vec{\eta}[s][a] + \alpha \left[r + \gamma Q_{(s',a')}(x') - Q_{(s,a)}(x)\right] \vec{e}[s][a].$$

* Update hyperstate and action:

$$\begin{array}{rcl} (s, x) & \leftarrow & (s', x') \\ a & \leftarrow & a'. \end{array}$$

### 4.3.2 Example

We apply this version of SARSA($\lambda$) to the BAMDP depicted in Figures 4.6 and 4.7, in which prior uncertainty for unknown transition probabilities is uniform ($\gamma = 1$, HORIZON=25, $\alpha = .05$, $\lambda = .85$, $\epsilon = .1$). The performance of the policy greedy with respect to evolving Q-values is plotted in Figure 4.9. As with our presentation of earlier results for the policy-gradient algorithm, each point plotted represents an estimate of expected total reward under the evolving policy—each point is the sample average, over 100,000 Monte-Carlo trials, of total reward obtained for a given policy.

## 4.4 Design for an optimal probe

Suppose we are confronted with a Markov decision process that has unknown transition probabilities, $p_{ij}^k$, $i, j = 1, 2, \ldots, N$, $k = 1, 2, \ldots, M$. We wish to identify these unknown parameters. At our disposal we have a "probe," which we can launch

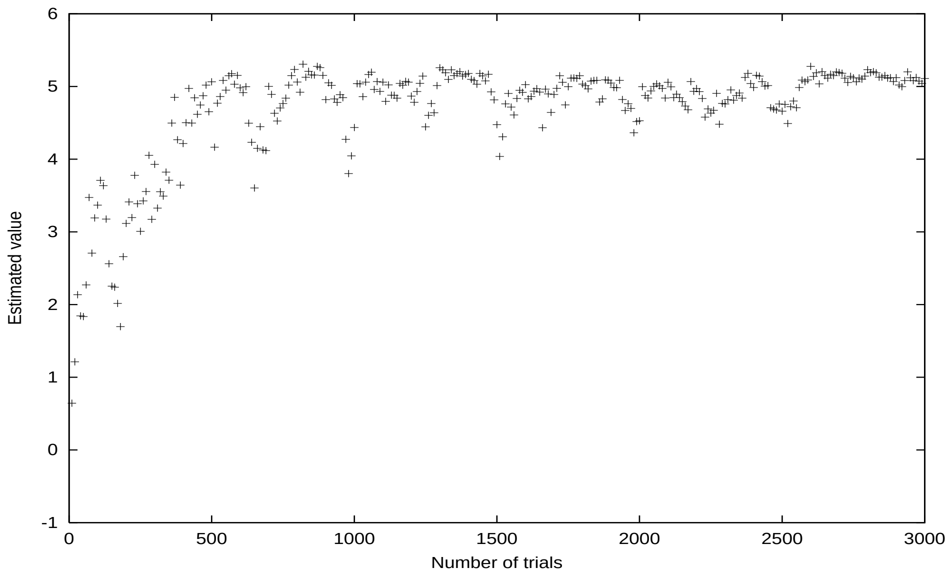


Figure 4.9: Performance of SARSA($\lambda$), with linear function approximators for Q-values, on the 2-state / 2-action example, with uniform priors for unknown transition probabilities ($\alpha = .05$, $\lambda = .85$).

into the MDP. The probe has a finite lifetime, LIFETIME, and may be pre-loaded with an adaptive policy for navigating through the MDP. Its goal is to reduce uncertainty as much as possible during its lifetime.

To make matters concrete, consider the specific example depicted in Figure 4.10, an MDP with seven states and two actions. ACTION 1 tends to move the probe left, while ACTION 2 tends to move the probe to the right.

We express our prior uncertainty in state-transition probabilities under these actions in terms of beta distribution parameters, which in Figure 4.10 label arcs. For example, our uncertainty in transitions from STATE 2 under ACTION 1 is defined by the beta-distribution parameter pair $(6, 2)$, so that the uncertainty in the probability of moving left or right is as depicted in Figure 4.11 (see Degroot (1970) for details of Bayesian revision of beta distributions with respect to Bernoulli sampling, *etc.*). Uncertainty distributions for the probability of moving left for all actions and states are shown in Figure 4.12.

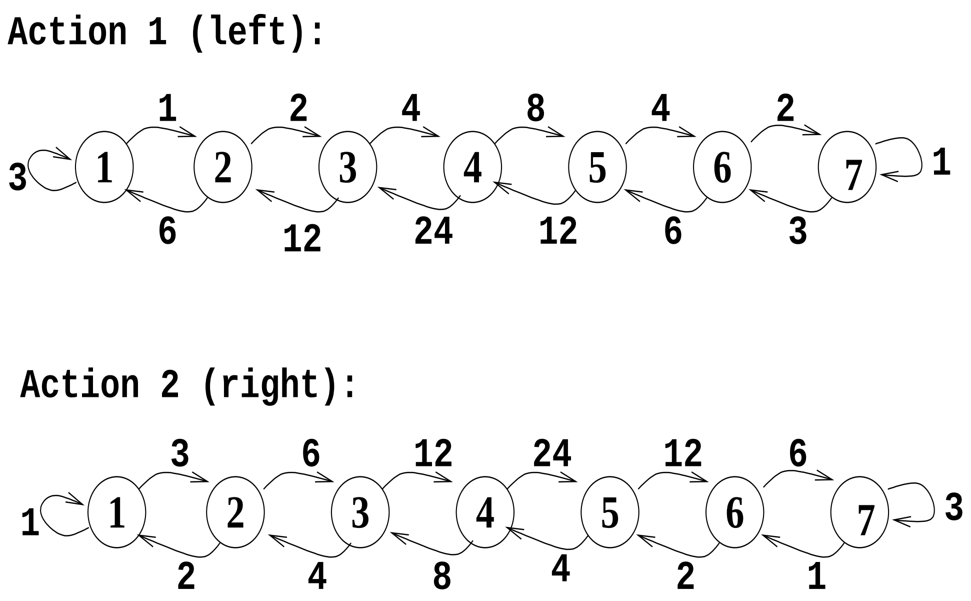


Figure 4.10: Beta distribution parameters label transition arcs for a simple optimal probing problem. Under ACTION 1, the process moves left with a mean probability .75, but with uncertainty that increases with distance from the middle (start) state. Under ACTION 2, the process moves *right* with mean probability .75, and uncertainty increases in the same way as for ACTION 1.

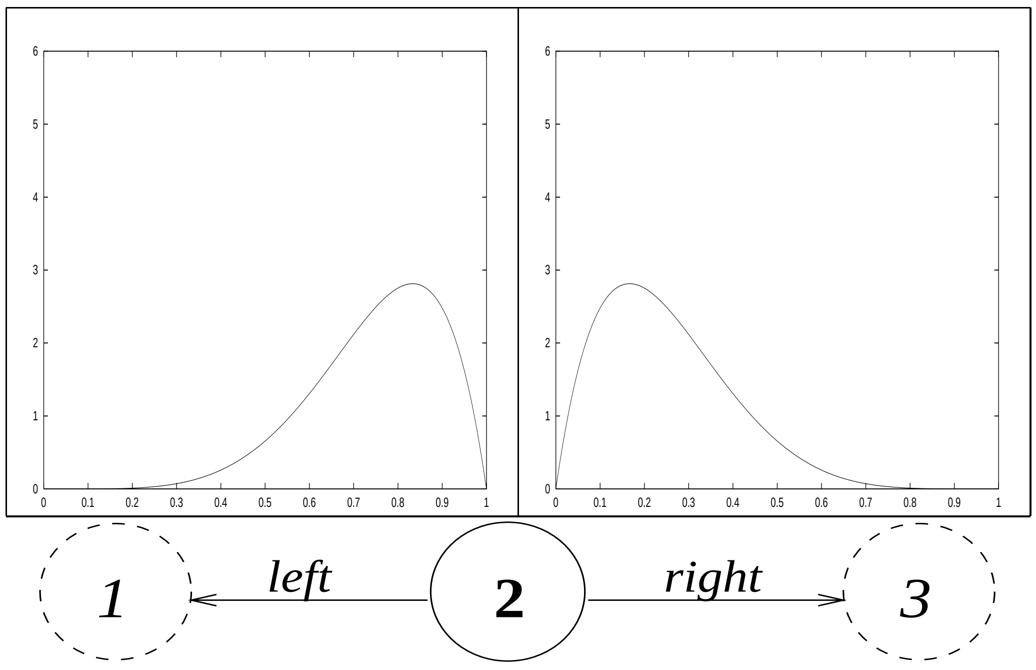


Figure 4.11: Uncertainty in state transitions from STATE 2 under ACTION 1 defined by the beta-distribution parameter pair $(6, 2)$. (The marginal prior distributions are shown plotted above their respective transitions; note that since $Pr\{\text{right}\} = 1 - Pr\{\text{left}\}$, one of priors defines uncertainty for both transitions.)

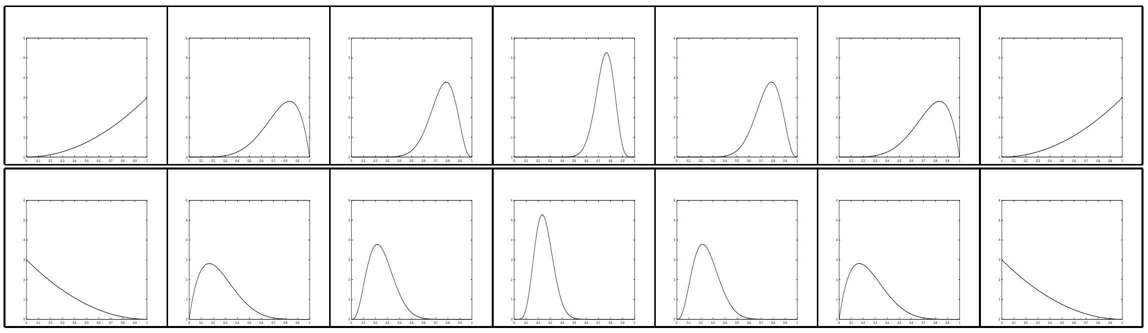


Figure 4.12: Prior uncertainty for the probability of moving left, for each state (STATE 1 through 7, left-to-right) for ACTION 1 (top row) and ACTION 2 (bottom row).

We have sculpted this example in such a way that our prior defines the *expected* probability of moving left under ACTION 1 as .75 for each state, but with uncertainty that increases with distance from the middle state, STATE 4, which we assume to be the start state. The prior uncertainty for ACTION 2 transitions is defined similarly (but with mean probability of moving *right* equal to .75).

We shall adopt the variance of the beta distribution as our measure of uncertainty:

$$Var(\alpha, \beta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

Let us denote the full set of beta-distribution parameters, which express our uncertainty in state transition probabilities, as $M = \{m_{ij}^k\}$ $i = 1, 2, \ldots, 7$, $k = 1, 2$, and $j \in \{L, R\}$, where "$L$" and "$R$" denote indices associated with left and right transitions, respectively. Thus 28 parameters suffice to specify our uncertainty in the MDP. The prior uncertainty may be assigned subjectively, or may be informed by the results reported from previous probes.

If we are in a particular state, $s$, and take action $a$, then experience a transition to the left, then our uncertainty in transition probability associated with state $s$ / action $a$ becomes:

$$Var(m_{sL}^a + 1, m_{sR}^a).$$

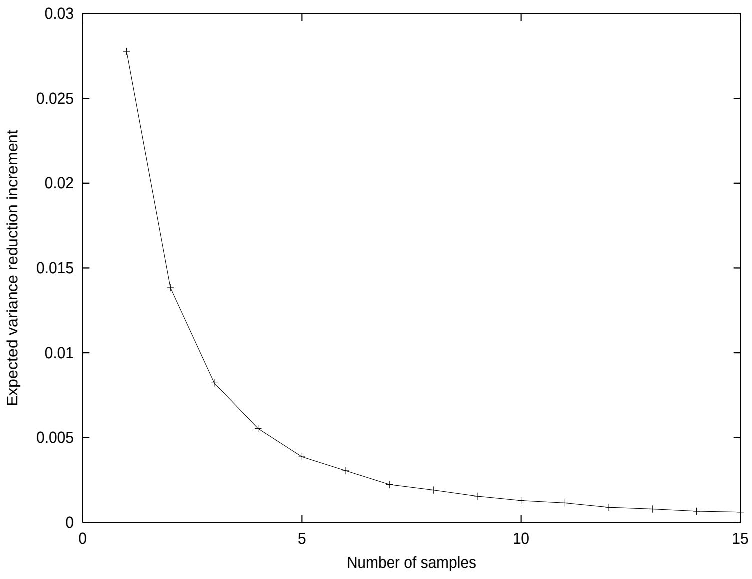


Figure 4.13: Diminishing variance reduction with repeated sampling. The graph plots the expected incremental reduction in variance for a Bernoulli process with unknown parameter and initial beta distribution parameters $(\alpha, \beta) = (1, 1)$.

We propose defining the reward associated with this transition as the reduction in variance; *i.e.,*

$$reward(s^a \to L|M) = Var(m^a_{sL}, m^a_{sR}) - Var(m^a_{sL} + 1, m^a_{sR}),$$

and the probe's goal is to maximize the expected sum of such rewards over its lifetime.[4]

We note that, if we repeatedly sample a fixed state and action, the expected reward (variance-reduction increment) diminishes with the number of samples (Figure 4.13). This, together with the assumed structure of prior uncertainty, should lead to non-trivial probing strategies. For example, one would think that the probe should drive toward one of the ends of the chain where variance is high, squash out uncertainty there, then at some point turn toward the opposite end, passing through low-variance states en route.

[4]Other definitions of uncertainty (*e.g.*, differential entropy) and reward are possible. At some level of abstraction, differential entropy measures may be related to our variance measure by appealing to the theoretical relationship that exists between differential entropy and Fisher information, which may be framed in terms of the respective volume and surface area of the "typical set" (Cover & Thomas, 1991).

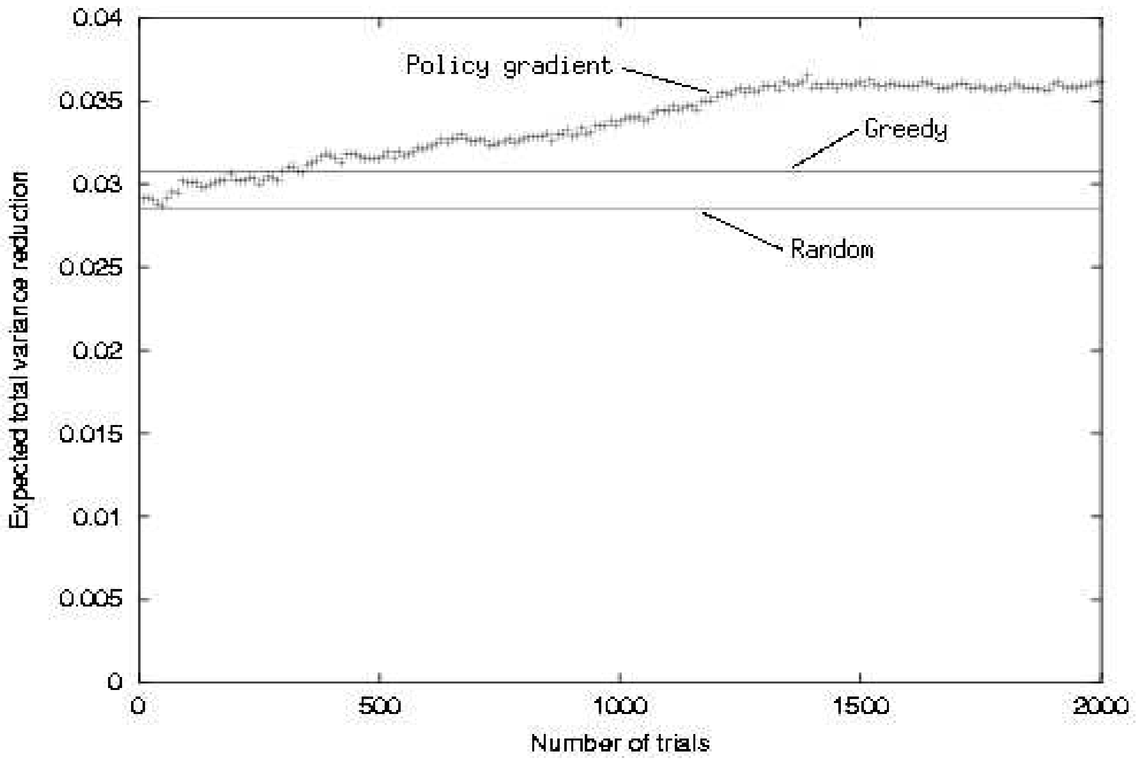


Figure 4.14: Probe performance (LIFETIME= 15).

We applied our policy-gradient algorithm to this BAMDP (LIFETIME = 15) and its performance is plotted in Figure 4.14 (each point is the expected performance as gauged by the average of 10,000 Monte-Carlo trials). As a comparison, we also show the expected performance of a probe that chooses its actions randomly, and the performance of a greedy probe that, at each step, chooses the action that maximizes the expected one-step reduction in variance. The policy-gradient probe yields approximately a 23% improvement over the random probe and an 11% improvement over the greedy probe.

## 4.5 Summary

Regarding the stochastic policy iteration algorithm presented in the first part of this chapter, more needs to be done to explore the relationship between the deterministic and stochastic algorithms' policy trajectories—in particular, whether there might be some advantage to allowing intermediate policies to be members of the more general feasible set of stochastic policies.

In fact, glancing at the various policy-trajectory plots, one wonders if there might be some advantage to incorporating a mechanism into our algorithms that would steer policies in a more direct fashion toward a solution—enhancements that would prolong a policy trajectory's occupancy of the inequality constraint set interior ("barrier" methods of nonlinear programming do this, as does Karmarkar's interior-point rescaling algorithm). Off-diagonal components of the gradient do not explicitly enter into the computation of a feasible direction of ascent $d$, but it is reasonable to suspect that they could play a role in "bending" $d$ toward an optimal solution. Offhand, it is not clear how the components $\frac{\partial V_{s'}}{\partial \pi_s^k}$, $s' \neq s$, should affect the action probabilities for state $s$.

For classical dynamic programming approaches, the complexity of computing optimal BAMDP policies scales exponentially with the time horizon. Even for two physical states and two physical actions, the HORIZON=25 case examined in our example is rapidly approaching the limits of tractability. However, the complexity of our simulation-based algorithm does not scale in this way. Space requirements are $O(N^3)$ for representing the value function and policy—space complexity is independent of the time horizon. It is difficult to be precise regarding the time complexity of our algorithm. We simply note that it is a trial-based algorithm, involving repeated estimation of $V^\pi$ through simulation of controlled trajectories of length equal to the time horizon. In *very* simple terms (we are neglecting the nonstationarity of the value function as the policy improves), estimation error scales inversely with the square root of the number of trials (and squared error is proportional to the variance), and each trial involves simulating a hyperstate trajectory, of length HORIZON, under the influence of the prevailing stochastic policy.

A TD($\lambda$)-based version of our algorithm with $\lambda = 1$ corresponds to strict Monte-Carlo estimation without "bootstrapping." This process provides unbiased estimates of the value function for a fixed policy, which for linear function approximation architectures implies stochastic-approximation style convergence of value-function parameters to values that minimize the mean-squared error of the value-function estimate (this holds for any distribution of backups). For $\lambda < 1$, TD($\lambda$) procedures utilize biased estimates of the value function (they perform bootstrapping), and in general they can not be guaranteed to converge to the mean-squared-error minimizing parameter values (though they have smaller variance and often perform better in practice than TD(1) methods).[5]

Sutton, McAllester, Singh, and Mansour (2000) and Konda and Tsitsiklis (2000) consider issues of convergence for actor-critic, policy-gradient algorithms of the type we have proposed and applied to BAMDP's.[6] One of their main conclusions is that features employed by the value function approximation architecture (the "critic") should span a subspace prescribed the choice of parameterization of the controller (the "actor")—the critic need only compute a certain projection of the value function (a high-dimensional object) onto a low-dimensional subspace spanned by a set of basis functions that are *completely* determined by the parameterization of the actor.[7]

[5]Let $\theta^*$ be the optimal parameter values, and let $\widetilde{\theta^*}$ denote the parameter values that TD($\lambda$) converges to. Then, it can be shown (Tsitsiklis & Van Roy, 1997) that

$$MSE\left(\widetilde{\theta^*}\right) = \frac{1-\gamma\lambda}{1-\gamma} MSE\left(\theta^*\right),$$

where $MSE(\theta) \equiv \sum_s \pi(s) \left[V^\pi(s) - V_\theta(s)\right]^2$, with $\pi$ understood to be the steady-state distribution associated with a fixed policy, $V^\pi$ the policy's true value, and $V_\theta$ the parameterized estimate. This result assumes that TD backups are distributed according to $\pi$; other backup distributions may actually lead to divergence.

[6]We note that gradient-based algorithms for reinforcement learning have a long history (*e.g.*, Werbos, 1977), and some approaches do not make use of value function estimates (Baxter & Bartlett, 2000; Marbach & Tsitsiklis, 2001, and Williams, 1992). A possible drawback of such "actor-only" methods is that the gradient estimates they produce may have large variance.

[7]The gradient of value with respect to controller parameters is the projection of Q-values onto the space spanned by gradients of state-action probabilities with respect to controller parameters. TD computes such a projection, and we only need to choose value-function features to adequately

In practice, if one assumes a particular parameterization of a policy, a "compatability condition" (Sutton *et al.*, 2000) suggests the functional form of parameterization for the value function approximator. For example, if we adopt an exponentiated, normalized functional form for stochastic policies, as we have done in Section 4.2.2, then the compatability condition would lead us to propose a linear architecture for the value-function approximator, which is exactly what we have done naively.

Since our algorithm is gradient-based, the best that one might hope for is that it would converge to a locally optimal policy—a policy for which the gradient of value with respect to controller parameters is zero. But if we use TD($\lambda$), with $\lambda < 1$, to adjust value function parameters, we have seen that we are guaranteed to converge to value-function parameter values that are only approximately optimal, and a statement regarding policy convergence must be weakened: the gradient of value with respect to controller parameters becomes "small infinitely often" (Konda & Tsitsiklis, 1999).

In summary, our actor-critic, policy-gradient algorithm for BAMDP's avoids the massive memory requirements inherent in conventional dyanamic programming (Bellman's "menace of the expanding grid"). The algorithm is practical and it works, at least to the extent that the function approximators invoked adequately represent value functions and policy mappings, and to the extent that the gradient scheme employed leads to improved policies. In contrast to many heuristics, the legitimacy of policies computed by the algorithm follows from the fact that the algorithm is grounded in Bellman's optimality equation defined over hyperstates—we acknowledge long-range effects of information gain by retaining the complete Bayesian formulation of the problem to be solved.

As mentioned in Chapter 1, one way of viewing the algorithm is that it is an offline computation of an online adaptive policy. Given a prior describing our uncer-

represent this projection.

tainty about an agent's environment, the algorithm "pre-compiles" a suitable adaptive strategy, which can be "loaded" into the agent and executed. In executing its adaptive strategy, the agent simply tracks the evolving hyperstate (*i.e.*, maintains state-transition counts) and applies actions as prescribed by its hyperstate-to-action policy mapping; that is, the agent is not required to perform significant amounts of computation online.

# CHAPTER 5

# ON THE CORRESPONDENCE BETWEEN PARTIALLY-OBSERVABLE MARKOV DECISION PROCESSES AND BAMDP'S

We have considered Bayes-adaptive Markov decision processes (BAMDP's) as models for decision making in which one adopts a Bayesian framework to model uncertainty in the transition probabilities associated with some underlying Markov decision process. Historically, researchers (Silver, 1963) have considered the "finite-parameter" or "multi-matrix" case in which, for example, it is assumed that the underlying Markov decision process is drawn from some finite set of possible generalized transition matrices that are assumed to be known, and, in this instance, one can cast the process straightforwardly as a classical partially-observable Markov decision process (POMDP). Here we attempt to extend the correspondence to the case where our uncertainty is expressed more generally in terms of distributions over transition probabilities, rather than over some some finite set of known matrices.

We begin this chapter by considering how, at a suitable level of abstraction, BAMDP's can be cast as POMDP's. We show how piecewise-linearity of the value function for partially-observable processes generalizes to a similar property for the

Bayes-adaptive case. For finitely-transient POMDP's, the optimal infinite-horizon value function is piecewise linear, and the optimal policy may be represented by a policy graph (or finite-state automaton). We explicitly consider such policies—we review how the associated value functions can be computed for POMDP's, and we carry out a similar investigation for BAMDP's. Finally, motivated by the BAMDP/POMDP correspondence, we adopt policies expressed as finite-state *stochastic* automata, and we propose a policy-improvement algorithm (similar in spirit to the policy-gradient algorithm presented in Chapter 4) that utilizes Monte-Carlo techniques for gradient estimation and ascent.

## 5.1 Partially-observable Markov decision processes

We begin this chapter with a review of the basic elements of partially-observable Markov decision processes, focusing on the development of a recursive procedure for expressing the optimal value function in terms of a finite set of vectors.

### 5.1.1 Dynamics

A partially-observable Markov decision process (POMDP) includes the following elements:

- A collection of state-transition probabilities,

$$p_{ij}^a \stackrel{\text{def}}{=} Pr\left\{S(t+1) = j | S(t) = i, a(t) = a\right\},$$

  just as for a conventional Markov decision process.

- A collection of conditional observation probabilities:

$$q_{jo}^a \stackrel{\text{def}}{=} Pr\left\{O(t+1) = o | S(t+1) = j, a(t) = a\right\}.$$

- A set of immediate rewards, $r^a_{ijo}$, which signify the rewards for transitioning from state $i$ to state $j$ under the influence of action $a$, and observing $o$. Using the state-transition and conditional observation probabilities, we may write

$$r^a_i \stackrel{\text{def}}{=} \sum_j p^a_{ij} \sum_o q^a_{jo} r^a_{ijo}.$$

In this framework, there exists an underlying Markov decision process but we do not have direct access to its physical state. Instead, for each action and successor state, there exists a probability distribution over possible observations. State, action, and observation sets here are all finite. At each time step, the process is in some physical state, say $i$, and the decision maker selects an action, say $a$. The process then makes a Markov transition to state $j$ with probability $p^a_{ij}$ and generates an observation $o$ with probability $q^a_{jo}$. The transition also produces an immediate reward, $r^a_{ijo}$, whose expected value, given that we are in state $i$ and apply action $a$, is $r^a_i$.

### 5.1.2 Belief state and Bayes revision

The information state, or "belief state," expresses our uncertainty regarding the identity of the current underlying Markov chain state:

$$\pi_i \stackrel{\text{def}}{=} Pr\{S(t) = i\} \qquad i = 1, ..., N.$$

Now consider how Bayes's rule prescribes how $\pi$ should be revised in light of an observation elicited by applying action $a$. The prior for $S(t+1)$ is given by:

$$\begin{aligned} Pr\{S(t+1)|a(t) = a\} &= \textstyle\sum_i Pr\{S(t+1) = j|S(t) = i, a(t) = a\}\, Pr\{S(t) = i\} \\ &= \textstyle\sum_i p^a_{ij}\pi_i. \end{aligned}$$

The likelihood function is just

$$Pr\{O(t+1) = o|S(t+1) = j, a(t) = a\} = q^a_{jo},$$

and Bayes's rule constructs the posterior as $\frac{\text{likelihood}\times\text{prior}}{\text{normalization}}$ :

$$Pr\left\{S(t+1)=j|a(t)=a,O(t+1)=o\right\}=\frac{q^a_{jo}\sum_i p^a_{ij}\pi_i}{\sum_j q^a_{jo}\sum_i p^a_{ij}\pi_i}.$$

We denote this prior-to-posterior transformation of belief state abstractly by

$$\pi'=T(\pi|a,o).$$

### 5.1.3 Bellman equation

The optimal value (expected discounted total reward) function with $t$ steps remaining is recursively defined in terms of the optimal value function with $t-1$ steps remaining:

$$V^t(\pi)=\max_a\left\{\sum_i \pi_i r^a_i+\gamma\sum_i \pi_i\sum_j p^a_{ij}\sum_o q^a_{jo}V^{t-1}\left(T(\pi|a,o)\right)\right\}.$$

It can be shown that $V$ is convex, and by induction that, for finite horizon problems, it is piecewise-linear. This means there exists some collection of vectors $\left\{\alpha^1_{(t)},\alpha^2_{(t)},...,\alpha^K_{(t)}\right\}$ such that:

$$V^t(\pi)=\max_k\left\{\pi\cdot\alpha^k_{(t)}\right\}.$$

The finite set of $\alpha$-vectors serves to represent the value function, which is a piecewise-linear, faceted surface defined over the continuous set of belief states.[1]

### 5.1.4 $\left\{\boldsymbol{\alpha}_{(t)}\right\}$ from $\left\{\boldsymbol{\alpha}_{(t-1)}\right\}$

Given that we have defined $V^{t-1}$ in terms of the set $\left\{\alpha^k_{(t-1)}\right\}^{K'}_{k=1}$, Bayes's rule and Bellman's equation determine a set of $\left\{\alpha^k_{(t)}\right\}^{K}_{k=1}$ that define $V^t$. First, applying

[1]Let $R^0(i)$, $i=1,...,N$ be terminal rewards or values assigned to terminal physical states. Then $V^0(\pi)=\sum_{i=1}^N \pi_i R^0(i)$—so $\alpha_{(0)}(i)$ can be identified with $R^0(i)$ $\forall i$. This proves piecewise-linearity for the base case. The inductive step now follows in the main text.

Bayes's rule,

$$\begin{aligned} V^{t-1}(\pi') &= \max_k \left\{ \sum_j \pi'_j \alpha^k_{(t-1)}(j) \right\} \\ &= \max_k \left\{ \sum_j \frac{q^a_{jo} \sum_i p^a_{ij} \pi_i}{\sum_j q^a_{jo} \sum_i p^a_{ij} \pi_i} \alpha^k_{(t-1)}(j) \right\}. \end{aligned}$$

In performing the $\max_k$ operation, the denominator is irrelevant. Let

$$k^*(\pi, a, o) \stackrel{\text{def}}{=} \arg\max_k \sum_j q^a_{jo} \sum_i p^a_{ij} \pi_i \alpha^k_{(t-1)}(j)$$

be the identity (index) of the segment of $V^{t-1}$ that $\pi$ gets mapped to under the transformation $\pi' = T(\pi | a, o)$. Now we can rewrite $V^{t-1}$ as:

$$V^{t-1}(\pi') = \sum_j \frac{q^a_{jo} \sum_i p^a_{ij} \pi_i}{\sum_j q^a_{jo} \sum_i p^a_{ij} \pi_i} \alpha^{k^*(\pi,a,o)}_{(t-1)}(j).$$

Substitution of this expression into Bellman's equation results in:

$$V^t(\pi) = \max_a \left\{ \sum_i \pi_i \underbrace{\left[ r^a_i + \sum_j \sum_o p^a_{ij} q^a_{jo} \alpha^{k^*(\pi,a,o)}_{(t-1)}(j) \right]}_{\alpha_{(t)}(i)} \right\}, \tag{5.1}$$

where the term within brackets, evaluated at $\arg\max_a$, defines an $\alpha_{(t)}(i)$; *i.e.*, a local facet or piece of $V^t$ defined in a neighborhood of $\pi$. We also associate the $\arg\max_a$ action with this $\alpha$.

### 5.1.5 Monahan's algorithm

Bellman's equation provides a way of determining the value-function facet associated with a particular belief state, but the number of belief states is uncountably infinite. If we could somehow substitute all possible belief states into Bellman's equation and solve it, we would arrive at a finite set of $\alpha$'s whose associated envelope would define the value function. Suppose we approach things from a different direction.

Note that there is only a finite number of candidate $\alpha$'s—they correspond to different choices for $a$ and $k^*$ in the bracketed term of Bellman's equation. Suppose the cardinality of the $\alpha_{(t-1)}$ set is $K'$, then the number of possible choices for $k^*$ is $K'$. The sum over $o$, with each term selecting any of the $K'$ $k^*$'s, allows for $K'^{|O|}$possibilities. Finally, suppose there are $A$ choices for action $a$; this implies a total of $AK^{|O|}$ possible candidate $\alpha_{(t)}$ vectors.

Monahan's algorithm (Monahan, 1982) generates all these candidate $\alpha_{(t)}$ vectors, then prunes the set using linear programming to find the maximizing envelope; *i.e.*, for each $\alpha^j_{(t)}$ candidate, we check to see if the linear programming problem with constraints,

$$\begin{aligned} \textstyle\sum_i \pi_i \alpha^k_{(t)}(i) &\leq \textstyle\sum_i \pi_i \alpha^j_{(t)}(i) \quad \forall k \\ \textstyle\sum_i \pi_i &= 1 \\ \pi_i &\geq 0 \quad \forall i, \end{aligned}$$

has a feasible solution (the objective function is irrelevant). If it does, then $\alpha^j$ defines part of the $V^t$ envelope. If it does not, then we can eliminate $\alpha^j$ from further consideration. While Monahan's algorithm may not be particularly efficient, it demonstrates concretely how optimal policies, defined over a continuous belief space, can be computed using a finite amount of computation. Other algorithms have been proposed for solving POMDP's, but many of them are closely related to Monahan's approach and utilize linear programming in computing the POMDP value function.

### 5.1.6 The infinite horizon case; policy graphs

Consider maximizing infinite horizon discounted reward. If we proceed to generate $V^t$ for successive values of $t$, we find that after a certain number of iterations the value functions begin to converge. In general, the limiting value function is convex. In some cases ("finite transience"), the limiting value function is also piecewise-linear.

Belief states in the domain of a particular $\alpha_{(t)}$ element all get mapped, by $T(\pi|a, o)$, to the domain of a unique $\alpha_{(t-1)}$ element. We can construct a directed graph in which vertices correspond to $\alpha_{(t)}$-elements and their optimal actions. Arcs associated with possible observations are directed to $\alpha_{(t-1)}$-vertices in a way that reflects the $T(\pi|a, o)$ mapping. In the finite-transient case, for $t$ large enough, $\left\{\alpha_{(t)}^{k}\right\}_{k=1}^{K} = \left\{\alpha_{(t-1)}^{k}\right\}_{k=1}^{K'}$, and we can "collapse" the graph to form a finite-state controller. One advantage of the finite-state controller, in which policies are mappings of memory states to actions, is that we need not keep track of the information state when controlling the system, we simply traverse the graph in accordance with the actions and observations associated with the vertices and arcs, respectively.

## 5.2 Bayes-adaptive Markov decision processes

For convenience, we repeat the mathematical characterization of BAMDP's that was presented earlier in Chapter 1, emphasizing the use of Bayes's rule and conjugate families of distributions, and we derive Bellman's equation in this context.

### 5.2.1 Transition matrix uncertainty

Define the *generalized transition matrix* as the matrix of transition probabilities $p_{ij}^{a}$ :

$$P \stackrel{\text{def}}{=} \begin{bmatrix} p_{11}^{1} & p_{12}^{1} & \cdots & p_{1N}^{1} \\ p_{11}^{2} & p_{12}^{2} & \cdots & p_{1N}^{2} \\ \vdots & \vdots & & \vdots \\ p_{11}^{A_1} & p_{12}^{A_1} & \cdots & p_{1N}^{A_1} \\ p_{21}^{1} & p_{22}^{1} & \cdots & p_{2N}^{1} \\ p_{21}^{2} & p_{22}^{2} & \cdots & p_{2N}^{2} \\ \vdots & \vdots & & \vdots \\ p_{21}^{A_2} & p_{22}^{A_2} & \cdots & p_{2N}^{A_2} \\ \vdots & \vdots & & \vdots \\ p_{N1}^{1} & p_{N2}^{1} & \cdots & p_{NN}^{1} \\ p_{N1}^{2} & p_{N2}^{2} & \cdots & p_{NN}^{2} \\ \vdots & \vdots & & \vdots \\ p_{N1}^{A_N} & p_{N2}^{A_N} & \cdots & p_{NN}^{A_N} \end{bmatrix}.$$

$A_i$ is the number of admissible alternative actions when in state $i$, and let $A = \sum_i A_i$ be the number of rows of $P$. Similarly, define a matrix, $R$, specifying a set of one-step immediate rewards, $r_{ij}^{a}$. Together, $P$ and $R$ suffice to specify a Markov decision process.

Now suppose that $P$ is a random variable with prior distribution $H(P|\psi)$. $H$ is a function of the $A(N-1)$ independent elements of $P$, and the distribution parameter, $\psi$, is in general a point in a multidimensional Euclidean space.

If we sample the process (observe a transition, "$x$"), then we can update $H$ using Bayes's rule:

$$dH(P|\psi, x) = \frac{l(x|P)dH(P|\psi)}{\int_P l(x|P)dH(P|\psi)};$$

*i.e.*, the posterior density is proportional to the likelihood function (a conditional density) times the prior density.

### 5.2.2 Conjugate families of distributions

Let $\mathcal{H}$ be a family of distributions indexed by $\psi$. If $H(P|\psi) \in \mathcal{H}$, and $H(P|\psi, x) \in \mathcal{H}$ for every $x$ and $\psi$, then $\mathcal{H}$ is "closed" with respect to the sampling rule which determines $l(x|P)$. $\mathcal{H}$ is a *conjugate family of distributions.* We denote the mapping induced by Bayes's rule as $\psi' = T(\psi|x)$ or, for an observation of a state $i$ to state $j$ transition under action $a$,

$$\psi' = T_{ij}^a(\psi).$$

For Markov decision processes, the state transition samples one observes are governed by multinomial distributions, one for each row of $P$. It can be shown that an appropriate conjugate family of distributions in this case is the "matrix beta", or Dirichlet, distribution. With a parameter matrix $M \equiv \left\{m_{ij}^a\right\}$, the joint density is given by

$$f(P|M) = \eta(M) \prod_{i,j=1}^{N} \prod_{a=1}^{A_i} \left(p_{ij}^a\right)^{m_{ij}^a - 1},$$

where $\eta(M)$ is a normalizing constant. In this case, the Bayes update of distribution parameters in light of observation takes on a particularly simple form:

$$M' = T_{ij}^a(M) = M + \delta_{ij}^a.$$

Here we identify $\psi$ with the $A \times N$ matrix, $M$, of Dirichlet parameters, and $\delta_{ij}^a$ with a $A \times N$ matrix with a "1" in the position associated with the observation (i.e., row $\sum_{l=1}^{i-1} A_l + a$, column $j$) and is zero otherwise.

### 5.2.3 Bellman equation

We may regard $(i, M)$ as a generalized state, or "hyperstate," where $i$ is the physical state of the Markov chain and $M$ indexes our uncertainty in $P$. Let $V_i(M)$ be the maximum over all policies (which map hyperstates to actions) of the discounted infinite horizon sum of rewards starting in hyperstate $(i, M)$.

Given a transition from $i$ to $j$ under action $a$, the maximum of the posterior discounted reward is

$$r_{ij}^a + \gamma V_j \left( T_{ij}^a(M) \right).$$

The probability of this transition given action $a$, with respect to the prior, is

$$\bar{p}_{ij}^a(M) \stackrel{\text{def}}{=} \int_P p_{ij}^a dH(P|M).$$

(If $H$ is taken to be a Dirichlet distribution with parameter $M$, then it can be shown that $\bar{p}_{ij}^a(M) = \frac{m_{ij}^a}{\sum_j m_{ij}^a}$.) The expected immediate reward given action $a$, with respect to the prior is

$$\bar{r}_i^a(M) = \sum_j \bar{p}_{ij}^a(M) r_{ij}^a.$$

Bellman's equation may be written as

$$V_i(M) = \max_a \left\{ \bar{r}_i^a(M) + \gamma \sum_j \bar{p}_{ij}^a(M) V_j \left( T_{ij}^a(M) \right) \right\}.$$

As with standard MDP's, Bellman's equation could serve as a basis for a successive approximation scheme for computing the optimal value function:

$$V_i^t(M) = \max_a \left\{ \bar{r}_i^a(M) + \gamma \sum_j \bar{p}_{ij}^a(M) V_j^{t-1} \left( T_{ij}^a(M) \right) \right\}.$$

Note that $V^t(M)$ is not computed with respect to $V^{t-1}(M)$, but rather with respect to $V^{t-1}\left(T(M)\right)$.

## 5.3 BAMDP's as POMDP's

We shall now reconsider the BAMDP formulation, following the development of the recursive procedure derived for the partially-observable case in Section 5.1.4. This will lead us to a parallel characterization of optimal value functions for BAMDP's.

### 5.3.1 General remarks

For POMDP's, the true (unknown) state of nature is the identity of the underlying physical state, $s(t)$, which can change with each time-step. The true state of nature is a member of a finite set. For BAMDP's, the (unknown) state of nature is the *generalized transition matrix*, $P$, (a matrix of all transition probabilities, $p_{ij}^a$; $r_{ij}^a$ denotes the corresponding one-step rewards) which is constant; the true state of nature is a point in a compact subset of $\mathcal{R}^{AN}$(where $A$ is the number of actions and $N$ is the number of physical states).

For POMDP's, observations are values of $o(t+1)$, which are generated via $q_{jo}^a$, a distribution over observations given $(a(t), s(t+1))$. For BAMDP's, observations are state transitions, $i^a \to j$ (state $i$, action $a$, resulting in transition to state $j$) which are generated via $p_{ij}^a$, a distribution over $s(t+1)$ given $(s(t), a(t))$.

For POMDP's, the belief state, $\pi$, is a distribution (point mass-function) over $N$ possible physical states. For BAMDP's, the information state, $dH(P|M)$, is a distribution (density) over possible generalized transition matrices.

For POMDP's, Bayes's rule revises the prior in light of observation via:

$$Pr\left\{S(t+1)=j|a(t)=a, O(t+1)=o\right\} = \frac{q_{jo}^a \sum_i p_{ij}^a \pi_i}{\sum_j q_{jo}^a \sum_i p_{ij}^a \pi_i},$$

which we write abstractly as $\pi' = T(\pi|a,o)$.

For BAMDP's, the update in light of an observed state transition is:

$$dH(P|M, S(t)=i, a(t)=a, S(t+1)=j) = \frac{p_{ij}^a dH(P|M)}{\int_P p_{ij}^a dH(P|M)}.$$

If $H$ is a member of a conjugate family of distributions, then we write $M' = T_{ij}^a(M)$.

### 5.3.2 Characterization of the optimal value function

For POMDP's, the value function is convex and, for finite horizons, piecewise-linear. Let us consider the case for BAMDP's by pursuing a path parallel to the development for the POMDP case. Let $R^0(j,P)$ be a terminal reward or value assigned to the true state of nature, $(j,P)$. Then, given an information state, $dH(P|M)$, the expected value is:

$$V^0(j, dH(P|M)) = \int_P R^0(j,P)dH(P|M).$$

Define $\alpha_{(0)}(j,P) \overset{\text{def}}{=} R^0(j,P)$; for each state, $j$, it is a real-valued function of $P$.

Now suppose (inductive hypothesis) that we can write

$$V^{t-1}(j, dH(P|M') = \max_k \left\{ \int_P \alpha^k_{(t-1)}(j,P)dH(P|M') \right\}$$

for some finite set of functions, $\left\{\alpha^k_{(t-1)}(j,P)\right\}_{k=1}^{K'}$. As we vary $M'$, different $\alpha$-functions will have the largest inner product with the density, $dH$, which is parameterized by $M'$. We now consider Bayes's rule applied with respect to a generic observation, $i^a \to j$, and substitute for $dH(P|M')$ :

$$V^{t-1}(j, dH(P|M') = \max_k \left\{ \int_P \alpha^k_{(t-1)}(j,P) \frac{p^a_{ij} dH(P|M)}{\int_P p^a_{ij} dH(P|M)} \right\}.$$

The denominator is irrelevant to the max operation, so let

$$k^*(M,i,a,j) \equiv \arg\max_k \left\{ \int_P \alpha^k_{(t-1)}(j,P) p^a_{ij} dH(P|M) \right\}, \tag{5.2}$$

so that

$$V^{t-1}(j, dH(P|M') = \frac{\int_P \alpha^{k^*(M,i,a,j)}_{(t-1)}(j,P) p^a_{ij} dH(P|M)}{\int_P p^a_{ij} dH(P|M)}.$$

Recall Bellman's equation, which we write as:

$$V^t(i, dH(P|M)) = \max_a \left\{ \sum_j \bar{p}^a_{ij} \left[ r^a_{ij} + \gamma V^{t-1}(j, dH(P|M') \right] \right\}.$$

Substitution for $V^{t-1}$ and expansion of $\bar{p}^a_{ij}$ leads to:

$$\begin{aligned}
V^t(i, dH(P|M) &= \max\Big\{ \textstyle\int_P \sum_j p^a_{ij} r^a_{ij} dH(P|M) \\
&\qquad\qquad + \gamma \textstyle\sum_j \bar{p}^a_{ij} \frac{\int_P p^a_{ij} \alpha^{k^*(M,i,a,j)}_{(t-1)}(j,P) dH(P|M)}{\int_P p^a_{ij} dH(P|M)} \Big\} \\
&= \max\Big\{ \textstyle\int_P \sum_j p^a_{ij} r^a_{ij} dH(P|M) \\
&\qquad\qquad + \gamma \textstyle\sum_j \int_P p^a_{ij} \alpha^{k^*(M,i,a,j)}_{(t-1)}(j,P) dH(P|M) \Big\} \\
&= \max_a \left\{ \int_P \underbrace{\left[ \sum_j p^a_{ij} \left( r^a_{ij} + \gamma \alpha^{k^*(M,i,a,j)}_{(t-1)}(j,P) \right) \right]}_{\alpha_{(t)}(i,P)} dH(P|M) \right\},
\end{aligned} \tag{5.3}$$

where the term within brackets, evaluated at $\arg\max_a$, may be taken as a definition for an $\alpha_{(t)}$.

## 5.3.3 Discussion

This last equation defines $V^t(i, dH(P|M)$ about a local neighborhood of some nominally chosen information state. With regard to a hyperstate specified by $(i, M)$, for example, and the finite set of functions, $\left\{\alpha^k_{(t-1)}(j,P)\right\}^{K'}_{k=1}$, which define $V^{t-1}$, we first compute $k^*(M, i, a, j)$ using Equation 5.2 for all possible $a$ and $j$. The summation over $j$ in the bracketed term of Equation 5.3 "marginalizes" the term's dependence on $j$, and for fixed $i$ what remains is a function of $P$ for each choice of $a$. Choosing the $a$ that maximizes the inner product of the term with the density, $dH(P|M)$, fixes $a$—for fixed $i$ and $M$, the newly constructed $\alpha_{(t)}(i,P)$ is a mixture (over $j$) of functions of the form $r^a_{ij} + \gamma\alpha^{k^*}_{(t-1)}(j,P)$, where $a$ is the $\arg\max_a$ action for Equation 5.3. We may associate this $a$ with the newly-constructed $\alpha_{(t)}$.

Thus we see that the idea of characterizing the value function in terms of a finite set of elements generalizes from the POMDP case. The fundamental differences stem from the fact that, for BAMDP's, the information state is a continuous density—a function rather than a finite-dimensional vector—which calls for appropriate generalization of inner product (from summation to integration) and re-definition of $\alpha$ (from vector to function).

Can we parallel the development of Monahan's algorithm for BAMDP's? Let us attempt to generate all possible candidate $\alpha_{(t)}$-functions, and then consider how we might go about pruning this set of superfluous members.

For fixed $i$, there are $A_i$ possible choices for $a$. In the sum over $j$, each term could select any of the $K$ possibilities for $k^*$. This implies a total of $A_i K^N$ (or $AK^N$ total for all $i$) possible $\alpha_{(t)}(i,P)$-functions. A particular candidate, $\alpha^j$, is not superfluous if there exists some region of information-state space in which its inner product with the density dominates that of all the other $\alpha$'s (in that region of the domain, $\alpha^j$ defines $V$'s envelope). In other words, we wish to check whether there exists a feasible $M$ such that

$$\int_P \alpha^k_{(t)}(i,P)dH(P|M) \leq \int_P \alpha^j_{(t)}(i,P)dH(P|M) \quad \forall k.$$

This set of linear-functional inequalities generalizes the linear system of constraints for POMDP's, which were addressed using linear programming. The case for BAMDP's is more problematic. Recall that for the POMDP case, the belief state was subject to the additional constraints, $0 \leq \pi_i \leq 1 \ \forall i$. The equivalent set of constraints for BAMDP's is that $dH$ must be a valid probability density. If $H$ is a Dirichlet distribution parameterized by $M$, then the integral constraints shift to constraints upon the parameter $M$. For infinite horizon problems, the feasible $M$-set will be an "integer-lattice" embedded in the positive orthant of $\mathcal{R}^{\mathcal{AN}}$ shifted to account for the prior. The feasible set of parameters is unbounded. It may even be true that all $\alpha's$ are relevant; *i.e.*, for every $\alpha$-function candidate, there may be a part of information state space in which the inner product of $\alpha$ with the density dominates all other such inner-products.

## 5.4 Policies represented by finite-state controllers

Previously we noted that, for finitely-transient POMDP's, the optimal infinite-horizon value function is piecewise linear, and the optimal policy may be represented by a policy graph. Here, we explicitly consider such policies—we review how the associated value functions can be computed for POMDP's, and we carry out a similar investigation for BAMDP's.

### 5.4.1 Finite-state controllers

The elements of a finite-state controller are:

- A finite set of inputs, which we take here to be the set of observations.
- A finite set of actions, which we take here to be the action-set of a POMDP or BAMDP.
- A finite set of memory states, $Q$.
- A memory state update function, $\tau : Q \times O \to Q$.
- An output function, $\alpha : Q \to A$.
- A start state.

We may think of a finite-state controller as a directed graph with actions labeling vertices, which correspond to memory states, and with arcs (labeled by observations) directed to successor memory states in a way that reflects the memory-state update function.

### 5.4.2 Value function for POMDP's

Note that a policy represented by a finite-state controller explicitly maps a finite number of memory states to actions, rather than an uncountable number of information states to actions, and while it may not be capable of representing an optimal policy, we can approximate the optimal policy arbitrarily closely by increasing the number of memory states.

We may think of a POMDP governed by a finite-state controller as interacting automata that form a Markov chain with a state-space that is the cross product, $S \times Q$, of the underlying MDP with the finite-state controller. We can write an system of linear equations for the value function associated with this hybrid process:

$$V(i,q) = \sum_j p_{ij}^{\alpha(q)} \left[ r_{ij}^{\alpha(q)} + \gamma \sum_o q_{jo}^{\alpha(q)} V(j, \tau(q,o)) \right].$$

If the process is in state $(i,q)$, then the finite-state controller prescribes action $\alpha(q)$. The expected infinite-horizon discounted reward is the expected one-step immediate reward, $\sum_j p_{ij}^{\alpha(q)} r_{ij}^{\alpha(q)}$, plus the expected discounted value of the successor state, $\gamma \sum_j p_{ij}^{\alpha(q)} \sum_o q_{jo}^{\alpha(q)} V(j, \tau(q,o))$. The value associated with some belief state, $\pi$, is just the weighted sum of value function components:

$$V(\pi, q) = \sum_i \pi_i V(i,q),$$

which defines the value function as a piecewise-linear function over belief space. If we start the controller in the memory state that maximizes the value of the starting state, then we may write:

$$V(\pi) = \max_q \sum_i \pi_i V(i,q),$$

which is convex as well as piecewise-linear.

## 5.4.3 Value function for BAMDP's

Let us now proceed with a consideration of finite-state controllers and their associated value functions for the case of BAMDP's. We define finite-state controllers as before, the only difference being in the interpretation of the observation set. For BAMDP's the observation set is the set of possible state-transitions, $i^a \to j$.

Our development for POMDP's began by considering the value associated with being in controller state $q$ when the true state of nature (the true underlying physical state of the Markov chain) was $i$. Similarly, consider the value associated with being in controller state $q$ when the true state of nature (the true underlying transition probabilities of the Markov chain) is the generalized transition matrix, $P$. We may write:

$$V\left((i,P),q\right) = \sum_j p_{ij}^{\alpha(q)} \left[ r_{ij}^{\alpha(q)} + \gamma V\left((j,P),\tau(q,j)\right)\right]. \tag{5.4}$$

For every generalized transition matrix, $P$, this defines a linear system of equations that can be solved to define $|Q|$ $N$-dimensional vectors. Considering this system for all feasible $P$'s implicitly defines a matrix of functions $\{v_{iq}(P)\}$.

Now if a BAMDP governed by a finite-state controller finds itself in physical state $i$, the controller is in memory state $q$, and our uncertainty in transition probabilities is expressed via $dH(P|M)$, then the associated value is given by

$$V\left((i, dH(P|M)),q\right) = \int_P v_{iq}(P) dH(P|M), \tag{5.5}$$

and if we start the controller in the memory state that maximizes the value of the starting state, we may write

$$V\left(i, dH(P|M)\right) = \max_q \int_P v_{iq}(P) dH(P|M). \tag{5.6}$$

#### 5.4.3.1 A simple computation procedure

Consider the following simple procedure for computing an approximation for the value function associated with a BAMDP governed by a fixed finite-state controller:

- Repeatedly (until integration error is reduced to some desired level),
  - Sample $dH(P|M)$ to generate a representative $P$.

  - Solve the system of equations signified by Equation 5.4 to get $V\left((i,q),P\right)$, $i = 1, ..., N$, $q = 1, ..., |Q|$.
  - Use the values, $V\left((i,q),P\right)$, in a numerical method for evaluating the integral in Equation 5.5; for example, we could simply form the sample average of solutions to Equation 5.4.

- Find the maximizing controller state as indicated by Equation 5.6.

#### 5.4.3.2 Discussion

In this chapter so far, we have attempted to be somewhat precise in answering the very reasonable and natural question, "Can BAMDP's be formulated as POMDP's?" We have seen that the answer is, "No, not exactly," but once one adopts a slightly-abstract viewpoint, several aspects of POMDP analysis translate to the BAMDP case in almost a straightforward or even mechanical way. We have seen that one characteristic of POMDP's, namely that value functions are determined by a finite set of vectors and their finite-dimensional inner-products with belief state components, generalizes to the BAMDP case, in which value functions are determined by a finite set of functions and their (integral) inner-products with information state densities. One casualty of information state taking on the form of a function rather than a vector is that linear-programming-based algorithms for POMDP's generalize to systems of linear integral inequalities in the case of BAMDP's and there appears to be no easy way to make use of this fact. Perhaps a more promising approach would start by adopting finite state controllers to define policies. For POMDP's the value function associated with a finite-state controller can be computed by solving a system of linear equations. We concluded this section by deriving similar results for BAMDP's, and we proposed a simple algorithm that incorporates statistical sampling and numerical integration for computing approximate value functions.

The next natural step in this investigation would be to develop procedures for constructing improved finite-state controllers for BAMDP's. Research (Hansen, 1998) on this issue for POMDP's makes use of an interpretation of the dynamic programming update used in policy improvement as the transformation and improvement of a finite-state controller. The dynamic programming update is essentially a one-step lookahead (a one-step policy choice), and for the infinite-horizon discounted case, the update may be viewed as generating a set of vectors $\left\{\alpha^k(t)\right\}_{k=1}^{K}$ that can be associated with potential memory states of a finite-state controller. Comparing these vectors with the existing set, $\left\{\alpha^k(t-1)\right\}_{k=1}^{K'}$(which define an existing controller), leads to an improved controller: (1) New vectors that dominate old essentially replace them, (2) Non-dominant new vectors result in new memory states, and (3) Old memory states that are not reachable from the new set are pruned. This approach offers a profound advantage over classical procedures that translate between finite-state controllers and polyhedral-region-to-action mappings; however, it is unclear whether such an approach can be adapted to the BAMDP case. As with Monahan's algorithm, a key computational step is the comparison of vectors, testing for dominance of one kind or another, and this is done using linear programming. As we have seen, in the BAMDP case, consideration of one-step policy choices leads to a finite set of functions, and tests for dominance are formulated in terms of systems of linear integral inequalities.

Other recent approaches for POMDP's (Meuleau, Kim, Kaelbling, & Cassandra, 1999) directly search a set of restricted finite-state controllers for the globally-optimal deterministic policy or a locally-optimal stochastic policy. We have shown how it is possible to construct a cross-product MDP for a POMDP governed by a finite-state controller in which the state set is $\{(s,q), s \in S, q \in Q\}$ and the action set is $\{(a,q), a \in A, q \in Q\}$—the action component for the controller is the choice of next memory state. Meuleau *et al.* show how to formulate and solve a Bellman equation for optimal value in this case, and though the solution is not implementable as a finite-state controller (it requires knowledge of physical state), it does provide an upper bound on performance. By systematically assigning values to free controller parameters—by specifying the memory-state-to-action and memory-state-and-observation-to-memory-state mappings—one can enumerate all controllers of some fixed size (enumeration is essentially a branch-and-bound procedure using breadth-first search and the upper bound). This problem is NP-hard; to simplify matters one may impose constraints upon the structure of feasible controllers. One may also consider stochastic policies represented by finite state controllers in which actions are determined via distributions over actions defined for each memory state and memory state transitions are conditional distributions over memory state given memory state and observation. It is possible to compute the gradient of the value function with respect to the design parameters of a fixed stochastic policy, and this gradient can serve as the foundation for classical numerical nonlinear optimization schemes based upon steepest ascent.

This stochastic policy ascent approach would seem to be a promising direction to pursue with regard to BAMDP's. The main analytical job to be done is to derive an expression for the gradient of the value function of a BAMDP governed by a stochastic policy (parameterized by a finite-state controller) with respect to the policy parameters (*i.e.*, action distributions and memory-state transition distributions). We now proceed to do this.

## 5.5 Monte-Carlo algorithms for the improvement of finite-state stochastic controllers: Application to BAMDP's

In this section, motivated by the correspondence between BAMDP's and partially-observable Markov decision processes, we adopt policies expressed as finite-state stochastic automata, and we propose policy improvement algorithms that utilize Monte-Carlo techniques for gradient estimation and ascent. Given the parameterization of policies by general finite-state stochastic controllers, the controller does not explicity model or track evolving uncertainty distributions; it simply traverses its associated state-transition diagram in accordance with observed state transitions, generating actions en route.

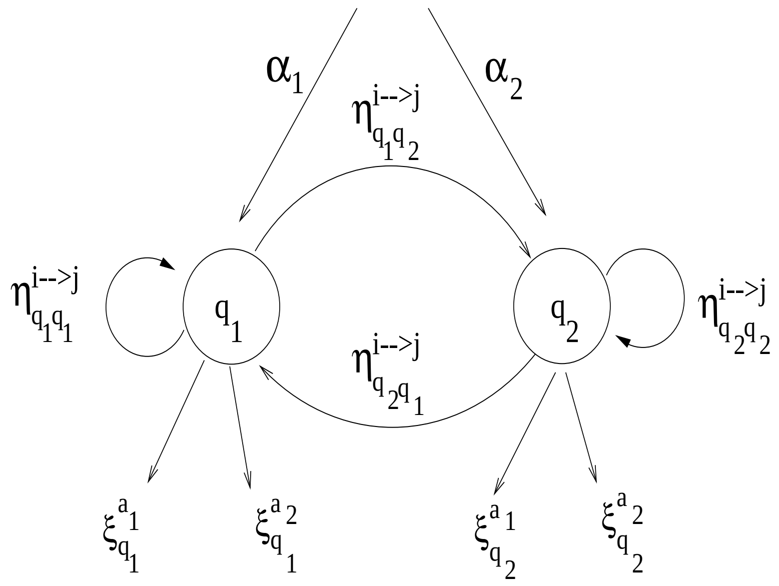
We first present general finite-state stochastic controllers, derive systems of linear equations for the value function and the value function gradient with respect to controller parameters, and propose a relatively direct Monte-Carlo algorithm for policy improvement. We conclude this chapter by proposing an alternative Monte-Carlo algorithm for gradient estimation that makes use of simulated process sample paths (this algorithm is similar in spirit to our policy-gradient approach presented in Chapter 4).

### 5.5.1 Finite-state stochastic controllers

The elements of a finite state stochastic controller are:

- A finite set, $\mathcal{Q}$, of memory states. (We use $Q$ to denote the *number* of memory states.)
- A finite set of inputs, which we take here to be the set of observable STATE$\rightarrow$ ACTION$\rightarrow$NEXT_STATE triples: $i^a \rightarrow j$.
- A distribution over starting memory states: $\alpha_q \overset{\text{def}}{=} Pr\{q_0 = q\}$.
- A distribution over actions for each memory state: $\xi_q^a \overset{\text{def}}{=} Pr\{a|q\}$.
- A memory state transition distribution for each memory state:

$$\eta_{qq'}^{i^a \rightarrow j} \overset{\text{def}}{=} Pr\left\{q'|q, i, j, a\right\}.$$



$\alpha_1$
$\alpha_2$
$\eta^{i\text{-->}j}_{q_1q_2}$
$\eta^{i\text{-->}j}_{q_1q_1}$
$q_1$
$q_2$
$\eta^{i\text{-->}j}_{q_2q_1}$
$\eta^{i\text{-->}j}_{q_2q_2}$
$\xi^{a_1}_{q_1}$
$\xi^{a_2}_{q_1}$
$\xi^{a_1}_{q_2}$
$\xi^{a_2}_{q_2}$

Figure 5.1: A simple finite state stochastic controller with two memory states (only one representative arc for each $\eta^{i^a \to j}_{qq'}$ $\forall i, j, a$ is shown for fixed $q$ and $q'$).

We may think of a finite-state stochastic controller as a directed graph with action-distributions associated with vertices, which correspond to memory states, and with arcs directed to successor memory states in a way that reflects the memory-state transition distributions (see Figure 5.1).

Note that a policy represented by a finite-state controller explicitly maps a finite number of memory states, rather than hyperstates (as in Chapter 4), to distributions over actions.

An MDP governed by a finite-state stochastic controller may be viewed as interacting automata that form a Markov chain with a state-space that is the cross product, $S \times \mathcal{Q}$, of the underlying MDP with the finite-state controller. The (discounted) value function associated with this hybrid process satisfies an equation expressing consistency with transitions to successor process states:

$$V\left((i,q)|P\right) = \sum_a \xi^a_q \sum_j p^a_{ij} \left[ r^a_{ij} + \gamma \sum_{q'} \eta^{i^a \to j}_{qq'} V\left((j,q')|P\right) \right] \quad i = 1, \ldots, N \quad q = 1, \ldots, Q. \tag{5.7}$$

This equation can be rewritten in the standard form, "$(I - \gamma A)x = b$," a linear system of equations in which $A$ has dimension $NQ \times NQ$, where

$$\begin{aligned} A_{(i-1)Q+q,(j-1)Q+q'} &= \textstyle\sum_a \xi^a_q p^a_{ij} r^a_{ij} \quad i,j = 1,\ldots,N \quad q,q' = 1,\ldots,Q \\ b_{(i-1)Q+q} &= \textstyle\sum_j \sum_a \xi^a_q p^a_{ij} r^a_{ij} \quad i = 1,\ldots,N \quad q = 1,\ldots,Q. \end{aligned} \tag{5.8}$$

Given that the Markov chain begins in state $i_0$, the expected value of the controlled process is

$$V_{i_0} = \sum_q \alpha_q V(i_0, q),$$

where, notationally, we understand that $V$'s value is for a fixed $P$.

### 5.5.2 Value gradient with respect to finite-state stochastic controller parameters

We shall employ exponentiated, normalized parameterized functions to represent all controller transition probabilities. This form is differentiable with respect to its parameters and ensures that the resulting functions define valid sets of transition probabilities for the controller. For example, we parameterize the initial controller memory state using $\{\phi_q\}_{q=1}^Q$ via $\alpha_q = \frac{e^{\phi_q}}{\sum_{q'} e^{\phi_{q'}}}$, $q = 1, ..., Q$. Similarly, action distributions are parameterized using $\left\{\chi^a_q\right\}$ via $\xi^a_q = \frac{e^{\chi^a_q}}{\sum_{a'} e^{\chi^{a'}_q}}$, $a = 1, ..., A \ \forall q$, and memory-state transition probabilities are parameterized using $\left\{\psi^{i^a \to j}_{qq'}\right\}$ via $\eta^{i^a \to j}_{qq'} = \frac{e^{\psi^{i^a \to j}_{qq'}}}{\sum_{q'} e^{\psi^{i^a \to j}_{qq'}}}$, $q' = 1, ..., Q \ \forall q, i, j, a$.

We now proceed to compute the gradient of $V_{i_0}$with respect to all controller parameters.

First, with regard to initial memory-state parameters, it can be shown that

$$\frac{\partial V_{i_0}}{\partial \phi_{\hat{q}}} = \alpha_{\hat{q}} \left[ V(i_0, \hat{q}) - \sum_q \alpha_q V(i_0, q) \right] \quad \hat{q} = 1, \ldots Q. \tag{5.9}$$

With regard to action parameters, starting from Equation 5.7, it can be shown that

$$\begin{aligned}\tfrac{\partial V(i,q)}{\partial \chi_{\hat{q}}^{\hat{a}}} \;&=\; \gamma \textstyle\sum_j \sum_{q'} \sum_a \xi_q^a p_{ij}^a \eta_{qq'}^{i^a \to j} \tfrac{\partial V(j,q')}{\partial \chi_{\hat{q}}^{\hat{a}}} \\ &\quad + \delta_{q\hat{q}} \xi_{\hat{q}}^{\hat{a}} \left\{ \textstyle\sum_j p_{ij}^{\hat{a}} \left[ r_{ij}^{\hat{a}} + \gamma \sum_{q'} \eta_{\hat{q}q'}^{i^{\hat{a}} \to j} V(j,q') \right] \right. \\ &\quad \left. - \textstyle\sum_a \xi_{\hat{q}}^a \left( \sum_{j'} p_{ij'}^a \left[ r_{ij'}^a + \gamma \sum_{q'} \eta_{\hat{q}q'}^{i^a \to j'} V(j',q') \right] \right) \right\},\end{aligned}$$

where $\delta_{q\hat{q}}$ is the Kronecker delta. Finally,

$$\frac{\partial V_{i_0}}{\partial \chi_{\hat{q}}^{\hat{a}}} = \sum_q \alpha_q \frac{\partial V(i_0,q)}{\partial \chi_{\hat{q}}^{\hat{a}}}.$$

With regard to memory-state transition parameters, it can be shown that

$$\begin{aligned}\tfrac{\partial V(i,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}\hat{a} \to \hat{j}}} \;&=\; \gamma \textstyle\sum_j \sum_{q'} \sum_a \xi_q^a p_{ij}^a \eta_{qq'}^{i^a \to j} \tfrac{\partial V(j,q')}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}\hat{a} \to \hat{j}}} \\ &\quad + \delta_{i\hat{i}} \delta_{q\hat{q}} \gamma \xi_{\hat{q}}^{\hat{a}} p_{\hat{i}\hat{j}}^{\hat{a}} \eta_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}} \left[ V(\hat{j},\widehat{q'}) - \textstyle\sum_{q'} \eta_{\hat{q}q'}^{\hat{i}^{\hat{a}} \to \hat{j}} V(\hat{j},q') \right],\end{aligned}$$

and

$$\frac{\partial V_{i_0}}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}\hat{a} \to \hat{j}}} = \sum_q \alpha_q \frac{\partial V(i_0,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}\hat{a} \to \hat{j}}}.$$

### 5.5.3 A direct Monte-Carlo policy improvement scheme

It can be seen that the formula for $\frac{\partial V(i,q)}{\partial \chi_{\hat{q}}^{\hat{a}}}$, for fixed $\hat{q}$ and $\hat{a}$, may be rewritten as a system of linear equations, "$(I - \gamma A)x = b$," where $A$ is the same matrix that appeared in linear system for $V(i,q)$, Equation 5.8, and $b$ is a vector that is zero but for $N$ elements corresponding to different choices of $i$; *i.e.*,

$$\begin{aligned}b_{(i-1)Q+\hat{q}} \;&=\; \xi_{\hat{q}}^{\hat{a}} \left\{ \textstyle\sum_j p_{ij}^{\hat{a}} \left[ r_{ij}^{\hat{a}} + \gamma \sum_{q'} \eta_{qq'}^{i^{\hat{a}} \to j} V(j,q' \right] \right. \\ &\quad \left. - \textstyle\sum_a \xi_{\hat{q}}^a \left( \sum_{j'} p_{ij'}^a \left[ r_{ij'}^a + \gamma \sum_{q'} \eta_{qq'}^{i^a \to j'} V(j',q) \right] \right) \right\} \qquad i = 1, \ldots, N.\end{aligned} \tag{5.10}$$

Similarly, the formula for $\frac{\partial V(i,q)}{\partial \psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$, for fixed $\hat{q}$, $\widehat{q'}$, $\hat{i}$, $\hat{j}$, and $\hat{a}$, may be written in the form "$(I - \gamma A)x = b$," where $A$ is as above and $b$ is a vector with one nonzero element:

$$b_{(\hat{i}-1)Q+\hat{q}} = \gamma \xi^{\hat{a}}_{\hat{q}} p^{\hat{a}}_{\hat{i}\hat{j}} \eta^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}} \left[ V(\hat{j}, \widehat{q'}) - \sum_{q'} \eta^{\hat{i}\hat{a}\to\hat{j}}_{\hat{q}q'} V(\hat{j}, q') \right] \tag{5.11}$$

To compute all the gradient components, we need only compute $A^{-1}$ once. We then multiply $A^{-1}$ by the appropriate $b$-vector to obtain the desired components. For example, to compute the gradient components with respect to initial memory state parameters, we begin by computing the value function components, $V(i_0, q)$, $q = 1, \ldots Q$, by multiplying the corresponding rows (rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$) of $A^{-1}$ by $b_V$, where $b_V$ denotes the $b$-vector associated with the value-function linear system given previously in Equation 5.8. $\frac{\partial V_{i_0}}{\partial \phi_{\hat{q}}}$ is then obtained from Equation 5.9.

To compute the gradient components with respect to action parameters, we begin by computing the gradient components, $\frac{\partial V(i_0,q)}{\partial \chi^{\hat{a}}_{\hat{q}}}$, $q = 1, \ldots, Q$, by multiplying rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$, of $A^{-1}$ by $b_{\chi^{\hat{a}}_{\hat{q}}}$, where $b_{\chi^{\hat{a}}_{\hat{q}}}$ denotes the $b$-vector associated with the action parameter gradient linear system given previously in Equation 5.10. Since $b_{\chi^{\hat{a}}_{\hat{q}}}$ contains only $N$ non-zero elements, only columns $(i-1)Q + \hat{q}$, $i = 1, \ldots, N$, of $A^{-1}$ contribute. Finally, we obtain $\frac{\partial V_{i_0}}{\partial \chi^{\hat{a}}_{\hat{q}}}$ as the $\alpha_q$-weighted sum of $\frac{\partial V(i_0,q)}{\partial \chi^{\hat{a}}_{\hat{q}}}$ components.

Similarly, to compute the gradient components with respect to memory-state transition parameters, we begin by computing the gradient components, $\frac{\partial V(i_0,q)}{\partial \psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$, $q = 1, \ldots, Q$, by multiplying rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$, of $A^{-1}$ by $b_{\psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$, where $b_{\psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$ is the $b$-vector associated with the memory-state transition parameter gradient linear system given previously in Equation 5.11. Since $b_{\psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$ contains only one non-zero element, only column $(\hat{i} - 1)Q + \hat{q}$, of $A^{-1}$ contributes. Finally, we obtain $\frac{\partial V_{i_0}}{\partial \psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$ as the $\alpha_q$-weighted sum of $\frac{\partial V(i_0,q)}{\partial \psi^{\hat{i}\hat{a}\to\hat{j}}_{\widehat{\hat{q}q'}}}$ components.

The preceding development has shown how we can compute the gradient of performance with respect to all controller parameters, given a particular value for the generalized transition matrix, $P$. Our ultimate goal is to compute a controller that is optimal with respect to the prior distribution over $P$, and a simple Monte-Carlo scheme repeatedly: (1) samples from this prior, (2) computes the value function and its gradient with respect to controller parameters, and (3) takes a small step in parameter space in the direction suggested by the exact gradient computed for the sample $P$.

This approach requires $O(N^2Q^2A)$ space to store the controller parameters. Inverting the $A$ matrix exactly, using LU-decomposition for example, is an $O(N^3Q^3)$ proposition, and is required each time we sample $P$. Alternatively, we could apply iterative matrix-inversion techniques, at the cost of $O(N^2Q^2)$ operations per iteration, to compute an approximate inverse. Given $A^{-1}$, computing the value function requires $O(NQ^2)$ multiplications per iteration. Then computing the gradients with respect to *all* initial memory-state, action, and memory-state transition parameters requires $O(Q^2)$, $O(N^2Q^2)$ , and $O(N^2Q^3A)$ multiplications, respectively. It is difficult to be precise regarding the time complexity of this algorithm. For a fixed policy, the squared-error of Monte-Carlo estimates is inversely proportional to number of samples (for gradient estimates, we note that squared-error is proportional to the variance of the gradient, which may be significant).

### 5.5.4 Monte-Carlo gradient estimation

We begin by noting that the matrix $A$ may be interpreted as the "policy-averaged" transition matrix associated with the hybrid MDP defined over $S \times Q$, and a Monte-Carlo approach for estimating the value function and its gradient with respect to controller parameters can make use of this interpretation. The linear systems presented in the previous section each have the form $(I - \gamma A)x = b$. Rewriting slightly, we have $x = (I - \gamma A)^{-1}b = \sum_{k=0}^{\infty}(\gamma A)^k b$.

The $(i_0, q_0)$th row of $\sum_k (\gamma A)^k$ may be interpreted as the expected (discounted) number of visits to hybrid process states given that we start the process in state $(i_0, q_0)$. For an episodic, undiscounted ($\gamma = 1$) problem, we can obtain an unbiased estimate for $x_{(i_0,q_0)}$ by starting the process in hybrid state $(i_0, q_0)$, then following a simulated hybrid process trajectory and accumulating the corresponding components of $b$; *i.e.*,

$$x_{(i_0,q_0)} \approx \sum_{(i,q)\in trajectory} b_{(i,q)}.$$

For discounted problems, we terminate each step of the simulated trajectory with probability $1 - \gamma$. Since the $A$ matrix is the same for all of the linear systems, we can use the same simulated trajectory to estimate the value function and all of its gradients as well. The foregoing interpretation suggests a Monte-Carlo algorithm of the following form:

- Initialize all controller parameters.
- Repeatedly,
  - Sample the prior distribution to obtain a generalized transition matrix, $P$.
    - Set the initial physical state to $i_0$, and sample the initial-memory state distribution, $\alpha$, to obtain $q_0$.
    - While the trajectory has not been terminated,
      - Call the current hybrid state $(i, q)$.
      - Sample $\xi$, $P$, and $\eta$ to obtain action $a$, next-state $i'$, and next-memory state $q'$ (and reward $r^a_{ii'}$).
      - Update the value function estimate; for instance, by performing a TD(0) update.

· Update the estimates of value gradient; various components of the $b$-vectors specified in Section 5.5.3 determine incremental contributions.

· Update the controller parameters by moving in the direction of the gradient.

· Let $(i, q) = (i', q')$, and (for discounted processes) terminate the trajectory with probability $1 - \gamma$.

Alternatively, instead of repeatedly sampling the prior distribution to obtain a generalized transition matrix, $P$, which in turn is sampled to generate physical-state transitions, we could repeatedly simulate BAMDP hyperstate trajectories as in Chapter 4, with the physical-state components of hyperstate transitions interpreted as physical-state transitions. Also, we could perform "batch" updates, waiting until trajectories terminate before making suitably scaled adjustments to estimates and controller parameters.

## 5.6 Summary

We began this chapter by considering how, at a suitable level of abstraction, BAMDP's can be cast as POMDP's. We showed how piecewise-linearity of the value function for partially-observable processes generalizes to a similar property for the Bayes-adaptive case. For finitely-transient POMDP's, the optimal infinite-horizon value function is piecewise linear, and the optimal policy may be represented by a policy graph (or finite-state automaton). We explicitly considered such policies—we reviewed how the associated value functions can be computed for POMDP's, and we carried out a similar investigation for BAMDP's. Finally, motivated by the BAMDP/POMDP correspondence, we adopted policies expressed as finite-state *stochastic* automata, and we proposed policy-improvement algorithms (similar in spirit to the policy-gradient algorithm presented in Chapter 4) that utilize Monte-Carlo techniques for gradient estimation and ascent.

Our survey of POMDP solution techniques has by no means been exhaustive, and there may well exist approaches that we have not considered that could be generalized and applied to the case of BAMDP's. Ideally, an approach based upon the adoption of finite-state controllers would not only incrementally improve the controller's parameters, but would also incrementally adapt, construct, or "grow," the controller topology. The work of Meuleau *et al.* (1999) may be viewed as doing this in a brute-force way via a direct search over a restricted set of finite-state controllers. Perhaps a more subtle approach could be based upon Hansen's algorithm for POMDP's (Hansen, 1998), in which dynamic programming's policy improvement update is interpreted as a transformation of a finite-state (deterministic) controller. How this interpretation can be extended to controllers represented by stochastic automata is an open problem.

# CHAPTER 6

# APPROXIMATE MEAN TRANSITION FREQUENCIES AND POLICY EVALUATION FOR BAYES-ADAPTIVE MARKOV CHAINS

If we are considering a BAMDP with a known reward structure and some fixed policy over a finite horizon, then the value of the process is simply the inner-product between the expected number of STATE→ACTION→STATE transitions of each kind with their associated expected immediate rewards. Questions about value thus reduce to questions about the distribution of transition-frequency counts. In this chapter, we address the problem of *analytically* determining the mean transition frequency counts associated with a Markov chain with unknown transition probabilities. We suggest a novel approach in which we model information-state components by diffusion processes, and acknowledge interdependencies between these components by defining a system of flux constraints that joint information-state trajectories must satisfy. Together, the diffusion models and flux-constraint system provide an analytical estimate for mean state-transition frequencies, and hence an approximate method for performing policy evaluation in the context of Bayesian MDP's.

## 6.1 Introduction

In our ongoing efforts to address what in reinforcement learning has come to be known as the "explore/exploit trade-off" we seek computational approaches that are firmly-grounded in Bellman's optimality equation. In this context, we model uncertainty in Markov chain transition probabilities in a Bayesian way; that is, probability distributions model the likelihood of true transition probability values, and these distributions evolve (via application of Bayes's rule) in response to observed physical Markov chain state transitions. Together, the physical Markov chain state and parameters defining uncertainty distributions (the *information state*) comprise the "hyperstate," which is a sufficient statistic for the process evolving under uncertainty, and a version of Bellman's equation for Markov chains (and for Markov decision processes) holds for the generalized process. A solution to Bellman's equation in this case defines the value-function as a function of hyperstate, and for Markov decision processes, the optimal value function implicitly defines the optimal adaptive learning policy, which maps hyperstates to optimal actions.

Our approach to the explore/exploit tradeoff directly attempts to approximate solutions to the hyperstate-version of Bellman's equation. Typically, researchers (Dearden *et al.*, 1998; Kaelbling, 1993; Meuleau & Bourgine, 1999; Thrun, 1992) dutifully note the existence of this equation, declare computation of its solution to be an intractable proposition, then move on to propose schemes that are based upon heuristics or considerations somewhat removed from Bellman's original equation, whose solution would in theory provide a complete and satisfying resolution of the explore/exploit trade-off issue.

That computing the solution to Bellman's equation defined over hyperstates is universally acknowledged as an intractable proposition is due to the fact that the set of reachable hyperstates grows exponentially with the time horizon. As we saw in Chapter 1, a solution procedure based upon dynamic programming must first enumerate the set of reachable hyperstates at the terminal horizon, then sweep backward from this terminal manifold, backing up values along the way to the initial hyperstate. For all but the most trivial of problems, the amount of memory required by this approach makes it infeasible.

In Chapter 4, we adopted techniques of reinforcement learning—-parametric representation of value functions and policies together with Monte Carlo sampling of hyperstate process trajectories—to derive a policy-gradient algorithm for computing approximately-optimal policies. This approach avoids the massive memory requirements inherent in conventional dynamic programming and it works, at least to the extent that the function approximators invoked adequately represent value functions and policy mappings, and to the extent that the gradient scheme employed leads to improved policies.

In this chapter, we pursue a different, more-analytic path. Our focus is on estimating the expected total-reward value of a Markov reward process (an MDP with a fixed policy), with unknown transition probabilities (whose uncertainty distributions are updated in a Bayesian way), over a finite (though arbitrarily long) time-horizon.

We note that if we are considering an MDP with a known reward structure, and our optimality criterion is to maximize expected total reward over some finite horizon, then the value of the process is simply the inner product between the expected number of STATE→ACTION→STATE transitions of each kind and their associated expected immediate rewards. Questions about value thus reduce to questions about the distribution of transition frequency counts.

Our analysis begins by examining the dynamics of information-state. For each physical Markov chain state, we view parameters describing uncertainty in transition probabilities associated with arcs leading from the state as that state's *information state component.* Information state components change when transitions from their corresponding physical state are observed, and on this time-scale, the dynamics of each information state component forms an embedded Markov chain. The dynamics of each embedded Markov chain is non-stationary but smoothly-varying. This allows us to introduce a diffusion model for the embedded dynamics, which yields a simple analytic approximation for describing the flow of information-state density.

The analysis of embedded information state component dynamics may thus be carried out independently for each physical-state component. In order to account for interdependency between components and statistically model their joint behavior, we incorporate flux constraints into our analysis; *i.e.*, we account for the fact that, roughly, the number of transitions into a physical state must be equal to the number of transitions out of that state. The introduction of flux constraints essentially links the local time-scales of each embedded component information-state process together in a way that accounts for the interdependence of joint information-state components and maintains consistency with the global time-scale that governs the true, overall hyperstate process.

In slightly more concrete terms, we model each embedded information-state component density by a Gaussian process. Incorporating this model into the flux constraints leads to a linear system of equations in which the coefficient matrix has random (normal) elements, and the expected value of the solution to this linear system provides an analytic approximation for the mean number of transition counts, and hence the expected value of the Markov reward process.

Analysis thus boils down to finding an approximation for the mean-value of the inverse of a matrix with random elements. In this chapter we show how this can be done when the matrix possesses the special structure implied by our problem, and in the final section, we briefly outline how our approach can be extended to decision processes.

## 6.2 Distribution theory

An analysis of the distribution of transition frequency counts associated with a Markov chain with unknown transition probabilities modeled in a Bayesian way has been conducted by Peter Whittle in a paper written nearly half a century ago (Whittle, 1955).

In this section, following Martin (1968), we review the mathematical framework leading up to Whittle's result, which we develop into a recursive scheme for computing mean values of frequency counts. As for Bellman's equation over hyperstates, an iterative implementation would require the enumeration of all reachable hyperstates, which is in general an intractable proposition.

The analytical details summarized in this section will not be used subsequently in this chapter, and so the reader unconcerned with Whittle's distributional theory may choose to skip to Section 6.3.

### 6.2.1 The Whittle distribution

Consider a Markov chain with known transition matrix $P$. Suppose we start the chain in state $u$ and observe $n$ transitions. Let $F$ be the transition count matrix associated with this sample of $n$ transitions; *i.e.*, $f_{ij}$ is the number of observed transitions from state $i$ to state $j$.

Let $f_{i\cdot} = \sum_{j=1}^{n} f_{ij}$, $i = 1, ..., N$, and $f_{\cdot j} = \sum_{i=1}^{n} f_{ij}$, $j = 1, ..., N$, be row and column sums of $F$, the number of transitions out of state $i$ and the number into state $j$, respectively. The elements of $F$ must adhere to the flux constraints:

$$f_{i\cdot} - f_{\cdot i} = \delta_{iu} - \delta_{iv}, \qquad i = 1, ..., N,$$

where $u$ is the initial state and $v$ is the final state. If state $i$ is neither the initial nor the final state, then the number of transitions out of state $i$ must be equal to the number of transitions into state $i$, and the right-hand side of the flux-constraint equation is zero. If state $i$ is the initial state and not the final state, then the right-hand side is one, while if state $i$ is the final state and not the start state, the right-hand side is $-1$. Finally, if state $i$ is both the start state and the final state, the right-hand side is zero.

Let $\phi(u, n, P)$ denote the set of all possible transition counts $F$ that can be realized from a sample of $n$ consecutive transitions of a Markov chain with transition matrix $P$ and initial state $u$. Regard $\widetilde{F}$ as a random matrix with range set $\phi(u, n, P)$. $\widetilde{F}$ has the Whittle distribution with parameter $(u, n, P)$ if $\widetilde{F}$ has the joint probability mass function:

$$\begin{aligned} f_W(F|u,n,P) &= F^*_{vu} \frac{\prod_i f_{i\cdot}!}{\prod_{ij} f_{ij}!} \prod_{ij} p_{ij}^{f_{ij}} & \qquad F \in \phi(u,n,P) \\ &= 0 & \qquad \text{otherwise.} \end{aligned}$$

Here it is understood that $v$ is the unique solution to the flux-constraint equations, and $F^*_{vu}$ is the $(v, u)th$ cofactor[1] of the matrix $F^*$ defined by

$$\begin{aligned} f^*_{ij} &= \delta_{ij} - \frac{f_{ij}}{f_{i\cdot}}, & \qquad f_{i\cdot} > 0 \\ &= 0 & \qquad f_{i\cdot} = 0. \end{aligned}$$

$F^*$ may be interpreted as $I - \overline{P}$, where $\overline{P}$ is a matrix of maximum-likelihood estimates for transition probabilities based upon the observed transition counts $F$.

Operationally, one could compute the probability of observing a particular $F$ by first using $F$ to construct the matrix $F^*$, then by solving the flux-constraint equations for the final state $v$, which together with the start state $u$ specifies which cofactor of $F^*$ needs to be computed.

Whittle's original derivation (Whittle, 1955) of the formula for the distribution of $\widetilde{F}$ made use of a rather ingenious method of multiple contour integration and a generalization to several variables of Lagrange's expansion of an implicit function as a power series. Subsequent proofs (Dawson & Good, 1957) derived the result

[1]We recall that the $(v, u)th$ cofactor of $F^*$ is defined as $(-1)^{u+v}$ times the determinant of the matrix derived from $F^*$ by deleting its $u$th row and $v$th column.

from known theorems on the number of universal paths in an oriented linear graph (the "BEST" theorem, see de Bruijn & Aardenne-Ehrenfest, 1950). Billingsley (1961) provides a simple inductive proof that the formula is correct, but supplies little insight into the formula's origins. We simply note that, without the $F^*_{vu}$ term, Whittle's formula resembles that based on a set of independent multinomial trials.

### 6.2.2 Beta-Whittle distribution

#### 6.2.2.1 The beta distribution

A random variable $X$ has a beta distribution with positive parameters $m_1$ and $m_2$ if $X$ has an absolutely continuous distribution whose probability density function is

$$\begin{aligned} f_\beta(x|m_1, m_2) &= \tfrac{\Gamma(m_1+m_2)}{\Gamma(m_1)\Gamma(m_2)} x^{m_1-1}(1-x)^{m_2-1} && 0 < x < 1 \\ &= 0 && \text{otherwise.} \end{aligned}$$

The gamma function $\Gamma(\cdot)$ is defined as

$$\Gamma(m) = \int_0^\infty x^{m-1} e^{-x} dx \qquad m > 0,$$

which for positive integer arguments reduces to $\Gamma(m) = (m-1)!$.

It can be shown that the mean and variance of $X$ are

$$E(X) = \frac{m_1}{m_1 + m_2}$$

and

$$Var(X) = \frac{m_1 m_2}{(m_1+m_2)^2(m_1+m_2+1)}.$$

The Beta distribution is of interest to us because the family of Beta distributions is a *conjugate family* for samples from a Bernoulli distribution. That is, suppose that $X_1, X_2, ..., X_n$ is a random sample from a Bernoulli distribution with an unknown value of the parameter $\theta$ (probability that $X_i = 1$, else zero). Suppose also that the prior distribution for $\theta$ is a beta distribution with parameters $m_1$ and $m_2$. Then the posterior distribution of $\theta$ when $X_i = x_i$, $i = 1, ..., n$ is a beta distribution with parameters $m_1 + \sum_{i=1}^{n} x_i$ and $m_2 + n - \sum_{i=1}^{n} x_i$.

#### 6.2.2.2 The multivariate beta (or Dirichlet) distribution

The multivariate beta distribution is an extension of the beta distribution to N dimensions.

A random vector $X = (X_1, ..., X_N)'$ has a multivariate beta distribution with parametric vector $\overrightarrow{m} = (m_1, ..., m_N)'$ if the probability density function of $X$ satisfies the following: Let $\overrightarrow{x} = (x_1, ..., x_N)'$ be any point in $R^N$ such that $x_i > 0$ for $i = 1, ..., N$ and $\sum_{i=1}^{N} x_i = 1$. Then

$$f_D(\overrightarrow{x}|\overrightarrow{m}) = \frac{\Gamma(m_1 + \cdots + m_N)}{\Gamma(m_1) \cdots \Gamma(m_N)} x_1^{m_1 - 1} \cdots x_N^{m_N - 1},$$

and $f_D(\overrightarrow{x}|\overrightarrow{m}) = 0$ at any other point in $R^N$.

We note that the family of multivariate beta distributions is a conjugate family for observations that have a multinomial distribution.

#### 6.2.2.3 The matrix beta distribution

The $N \times N$ transition probability matrix $\widetilde{P}$ has the matrix beta distribution with parameter $M$ if $\widetilde{P}$ has the joint density function

$$f_{M\beta}(P|M) = k(M) \prod_{i=1}^{N} \prod_{j=1}^{N} p_{ij}^{m_{ij} - 1}$$

where the normalizing constant $k(M)$ is given by

$$k(M) = \prod_{i=1}^{N} \frac{\Gamma(m_{i \cdot})}{\prod_{j=1}^{N} \Gamma(m_{ij})}$$

with $m_{i \cdot} = \sum_{j=1}^{N} m_{ij}$.

The matrix beta density function is the product of $N$ multivariate beta density functions:

$$f_{M\beta}(P|M) = \prod_{i=1}^{N} f_D(\overrightarrow{p_i}|\overrightarrow{m_i}).$$

Let $\widetilde{P}$ have the matrix beta distribution with parameter $M$ and suppose that a sample of $n$ transitions yields an observed transition frequency count $F$. Then the conditional probability, given that $\widetilde{P} = P$, of observing $F$ is $\prod_{i=1}^{N}\prod_{j=1}^{N} p_{ij}^{f_{ij}}$. By Bayes's theorem the posterior density function is proportional to the product of the kernel of the likelihood function and the kernel of the prior density function. This implies that the posterior distribution of $\widetilde{P}$ is matrix beta with parameter $M' = M + F$.

#### 6.2.2.4 The beta-Whittle distribution

The beta-Whittle distribution is defined to be the unconditional distribution of the transition count $\widetilde{F}$ of a Markov chain with transition probability $P$, which is drawn from a matrix beta distribution.

Let $\phi(u, n)$ denote the set of all possible transition counts $F$ that could occur as a result of observing $n$ consecutive transitions of a Markov chain with initial state $u$ and a positive transition matrix. The beta-Whittle probability mass function with parameter $(u, n, M)$ is

$$f_{\beta W}(F|u, n, M) = \int f_W(F|u, n, P) f_{M\beta}(P|M) dP$$

for $F \in \phi(u, n)$. It is straightforward to show that

$$f_{\beta W}(F|u, n, M) = F^*_{vu} \frac{\prod_{i=1}^{N} f_{i\cdot} B(f_{i\cdot}, m_{i\cdot})}{\prod_{i=1}^{N}\prod_{j=1}^{N} f_{ij} B(f_{ij}, m_{ij})}$$

where $B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ is the beta function, and where $v$ is the unique solution to the flux-constraint equations provided previously.

### 6.2.3 Moments and their computation

The moments of the beta-Whittle distribution are somewhat complicated to compute. If the transition matrix $P$ were known, then the number of observed transitions from state $i$ to state $j$ in a trajectory of length $n$ starting in state $u$ has a Whittle distribution whose mean is given by

$$E(\widetilde{f}_{ij}) = \sum_{k=0}^{n-1} p_{ui}^{(k)} p_{ij},$$

where $p_{ui}^{(k)}$ denotes the $(u,i)th$ entry of $P^k$; that is, an interpretation of the summand is that the process travels from state $u$ to state $i$ in $k$ steps then makes a transition from state $i$ to state $j$. This equation can provide the basis for a recursive algorithm for computing the mean of the Whittle distribution. Alternatively, one can derive a spectral representation for the expected value of $\widetilde{f}_{ij}$ in terms of the eigenvalues of $P$.

If $\widetilde{F}$ has the beta-Whittle distribution with parameter $(u, n, M)$, then

$$E(\widetilde{f}_{ij}) = E_P E_{F|P}(\widetilde{f}_{ij}) = \sum_{k=0}^{n-1} E_P\left[\widetilde{p}_{ui}^{(k)} \widetilde{p}_{ij}\right].$$

A fundamental Lemma proved by Martin (1967, p. 21) states that if $H(\widetilde{P}|\psi)$ is a family of distributions closed under consecutive sampling of a Markov chain (*e.g.*, suppose $H$ is matrix beta with parameter $\psi = M$), and if $g(\widetilde{P})$ is any integrable function of $\widetilde{P}$, then

$$\int \widetilde{p}_{ij} g(\widetilde{P}) dH(\widetilde{P}|\psi) = \overline{p}_{ij}(\psi) \int g(\widetilde{P}) dH(\widetilde{P}|T_{ij}(\psi)),$$

where $\overline{p}_{ij}(\psi)$ is the marginal expectation of $\widetilde{p}_{ij}$ relative to the prior distribution $H(\widetilde{P}|\psi)$.

Applying the Lemma to our expression for $E(\widetilde{f}_{ij})$ yields

$$\begin{aligned} E_P(\widetilde{p}_{ij}\widetilde{p}_{ui}^{(k)}) &= \overline{p}_{ij}(M) \textstyle\int \widetilde{p}_{ui}^{(k)} dH(\widetilde{P}|T_{ij}(M)) \\ &= \overline{p}_{ij}(M) \overline{p}_{ui}^{(k)}(T_{ij}(M)), \end{aligned}$$

but

$$\begin{aligned}\overline{p}_{ui}(T_{ij}(M)) &= \textstyle\sum_{l=1}^{N} \int p_{ul}^{(k-1)} p_{li} dH(\tilde{P}|T_{ij}(M)) \\ &= \textstyle\sum_{l=1}^{N} \overline{p}_{li}(T_{ij}(M)) \int p_{ul}^{(k-1)} dH(\tilde{P}|T_{li}(T_{ij}(M))) \\ &= \textstyle\sum_{l=1}^{N} \overline{p}_{ul}^{(k-1)}(T_{li}(T_{ij}(M)))\overline{p}_{li}(T_{ij}(M)).\end{aligned}$$

These considerations lead to a recursive scheme for computing mean values of frequency counts. However, as for Bellman's equation over hyperstates, an iterative implementation would require the enumeration of all reachable hyperstates, which is in general an intractable proposition.

## 6.3 Analytic approximation

### 6.3.1 Diffusion approximation and solution

#### 6.3.1.1 Information drift of a Bernoulli process

Consider first a Bernoulli process with unknown parameter $\theta$ ("probability of success"). If we use a beta distribution to model the uncertainty in $\theta$, then two parameters suffice (the parameters are directly related to the number of successes and failures observed)—we choose the beta distribution to model uncertainty because it is a *conjugate family* for samples from a Bernoulli distribution. Recall from Section 6.2.2 that this means that if our prior uncertainty is described by a beta distribution specified by a particular pair of parameters, then our uncertainty after observing a Bernoulli process sample is also described by a beta distribution, but with parameter values that are only slightly (and very simply) adjusted—the beta distribution is "closed under Bernoulli sampling" (DeGroot, 1970). Figure 6.1 shows the state transition diagram for this elementary process. Each information state determines the transition probabilities leading out of it in a well-defined way.

In preparation for what follows, we now submit the information state space to a rotation and translation:

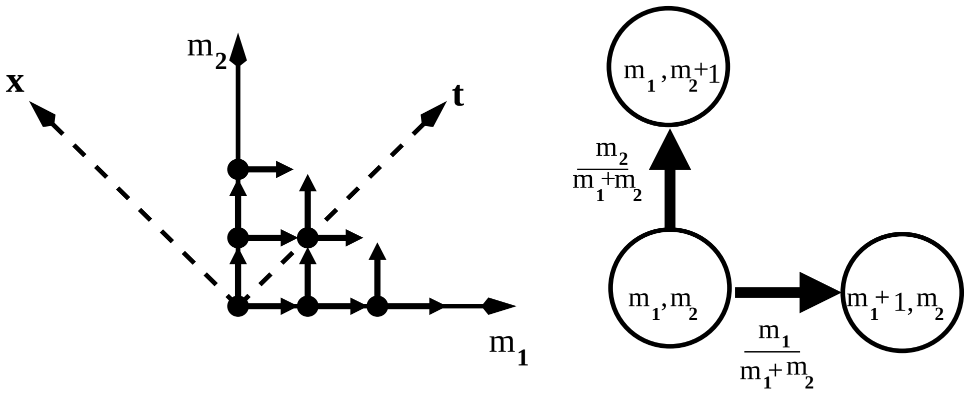


Figure 6.1: The state transition diagram associated with a Bayes-Bernoulli process. Transition probabilities label arcs in the exploded view. Dashed axes show the affine transformed coordinate system, whose origin coincides with $(m_1, m_2) = (1, 1)$ .

$$\begin{aligned} t &= m_1 + m_2 - 2 \\ x &= m_2 - m_1. \end{aligned}$$

In the new coordinate system, the probability of moving up or down $\Delta x = 1$ in time $\Delta t = 1$ is $\frac{1}{2}(1 + \frac{x}{t+2})$ and $\frac{1}{2}(1 - \frac{x}{t+2})$ respectively. And in analogy to the passage from random walks to Wiener processes, we propose a limiting process in which the probability of moving up or down $\Delta x$ in time $\Delta t$ is $\frac{1}{2}(1 + \frac{x}{t+2}\sqrt{\Delta t})$ and $\frac{1}{2}(1 - \frac{x}{t+2}\sqrt{\Delta t})$ respectively. Taking the limit as $\Delta t \to 0$ (with $\Delta x = \sqrt{\Delta t}$ ) results in a diffusion model with drift $\frac{x}{t+2}$ and diffusion coefficient 1; the density function of $x(t)$ is specified (by Kolmogorov's equations) as soon as these two local quantities have been determined. Alternatively, we may formulate the dynamics of $x$ in terms of the stochastic differential equation (SDE):

$$dx = \frac{x}{t+2}dt + dW_t,$$

where $W_t$ is a Wiener process and the symbolic differential may be rigorously interpreted as an (Ito) integral equation for each sample path.

This is a homogeneous equation with additive noise; it is "linear in the narrow sense" and an application of Ito's formula leads to the closed form solution:

$$x_t = \frac{t+2}{t_0+2} x_{t_0} + (t+2) \int_{t_0}^{t} \frac{1}{s+2} dW_s,$$

which implies that the solution is a Gaussian process. Its mean is

$$E(x_t) = x_{t_0} \frac{t+2}{t_0+2}.$$

The second moment, $E(x_t^2) = P(t)$, may be shown to satisfy $\frac{dP}{dt} = \frac{2}{t+2} P + 1$, which by introducing an integrating factor can be solved: $P(t) = c(t+2)^2 - (t+2)$, where $c = \frac{x_{t_0}^2 + (t_0+2)}{(t_0+2)^2}$.

Many steps have been omitted and we have deliberately refrained from attempting to review the relevant background material on second order processes and Ito calculus (Arnold, 1974; Kloeden & Platen, 1992). Our point is that diffusion models can be developed in this context and that they can yield useful information (namely, estimates for the mean and variance of the information state as a function of time).

#### 6.3.1.2 Information drift of a Bayes-adaptive Markov chain

The preceeding development has modeled the information state dynamics of an unknown Bernoulli process, but what we are really interested in is a similar model for Markov chains with unknown transition probabilities. The two are related. Consider Figure 6.2, which depicts a Markov chain with two physical states and unknown transition probabilities.

We may think of each physical state as having associated information state components that describe the uncertainty in the transition probabilities leading from that physical state—these information state components evolve only when the associated physical state experiences a transition, and on a time-scale defined by these particular physical state transitions, the evolution of the corresponding information

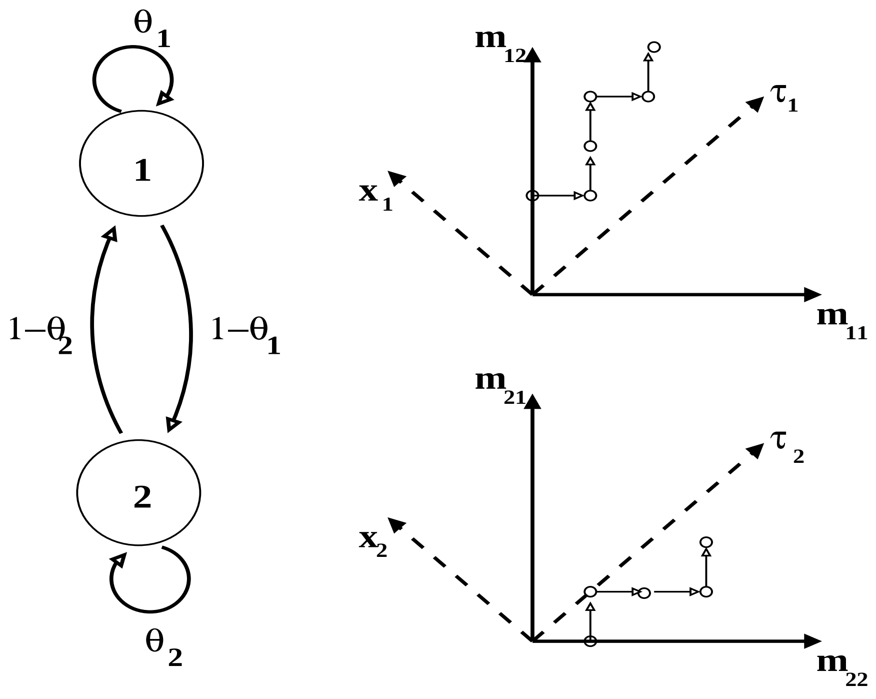


Figure 6.2: A Markov chain with two physical states and unknown transition probabilities, along with the component information state transition diagrams (the information state trajectory shown corresponds to the following sequence of physical state transitions: $1 \to 1 \to 2 \to 1 \to 2 \to 2 \to 2 \to 1 \to 1 \to 2$).

state components exactly follows that of a Bernoulli process with an unknown parameter. Moreover, the information state component transition diagrams associated with each physical state are identical; all information state components move about upon identical terrain—the only differences between the information state components' dynamics are: (1) they may have differing initial conditions (defined by their priors at starting time), and (2) their "local clocks" may tick at different time-varying rates—the matriculation rate of a given component of information state through its transition diagram is determined by the rate of physical state transitions observed from the information state component's corresponding physical state.

## 6.3.2 Flux constraints

In Section 6.3.1 we derived a diffusion model for the density function associated with embedded information state components. What we desire is a model for the *joint* density of these components under the constraint that they correspond to physical-state trajectories of a Markov chain. The information-state trajectories must conform to certain constraints of flux—the implied concerted flow into and out of physical states—in order to be consistent with realizable physical state trajectories. Later in this section, we shall argue that these flux constraints imply that reachable information states must lie within regions defined, in part, by certain particular hyperplanes in the joint information state space.

#### 6.3.2.1 Numerical SDE's and sample path approximation

We can numerically integrate information component SDE's to generate sample paths of information state trajectories, and then use the flux constraints to "retract" these trajectories so that together they correspond to a feasible physical state trajectory.

For the scalar stochastic differential equation,

$$dx_t = a(t, x_t)dt + b(t, x_t)dW_t,$$

the generalization of the Euler approximation for ordinary differential equations, in which the right hand side of the equation is "frozen" at its value at its left grid point on each time-step subinterval, is the *Euler-Maryama* approximation:

$$y_{n+1} = y_n + a(\tau_n, y_n)(\tau_{n+1} - \tau_n) + b(\tau_n, y_n)(W_{\tau_{n+1}} - W_{\tau_n}).$$

In practice, the Wiener process increment can be constructed by scaling a standard Gaussian pseudo-random number by $\sqrt{\Delta\tau}$ . For SDE's with additive noise, the approximation can be shown to have strong order of convergence equal to 1 (*i.e.*, its global discretization error is linear in $\Delta\tau$), which matches the error-order of Euler's method for deterministic equations. If we are given a set of component information state trajectories, $\left\{x^i(\tau^i), \tau_0^i \leq \tau^i \leq \tau_f^i\right\}_{i=1}^n$ and the initial physical state $s_{t_0}$, we can, in a straightforward manner construct the corresponding physical state trajectory. We simply follow the information state diffusion $x^{s_{t_0}}$ until it "level-crosses"; *i.e.*, it reaches a boundary line (or hyperplane in higher dimensions) that signifies a change in physical state. For example, in Figure 6.2, lines defined by constant integral values of $m_{12}$ signify boundaries that, when crossed by the information state process, indicate a physical state transition to state 2. We then follow the information state diffusion associated with the new physical state until it level-crosses, and so on.

This procedure constructs sample paths in a left-to-right fashion. An alternative approach starts from an initial set of prospective component information state trajectories and then performs a sequence of "retractions", in which a trajectory that violates the flux constraints is shortened; its $\tau_f$ value is moved backward in time until the flux constraint is satisfied. Different trajectories may be retracted at each step (the same trajectory may be retracted more than once).

To make matters more concrete, a set of information state trajectories determines a matrix $F$, which is used to compute $\text{FLUX}_s \overset{\text{def}}{=} f_{\cdot s} - f_{s\cdot} + I_{\{s_{t_0}=s\}}$ for each physical state $s$. We then retract the information state trajectory having minimum flux; we move backward along its trajectory, which results in decrementing its row of $F$, until its flux is 0. This process of retraction repeats until no states have negative flux (one state will have FLUX$= 1$, the others will have FLUX$= 0$). The *maximal* interpretation of the information state trajectories is one in which the trajectories have been retracted to their final levels, and the FLUX$=1$ state is actively diffusing toward its next level crossing.

Stochastic time-discrete approximation of the diffusion model provides an efficient and flexible means for generating component information state trajectories, and the retraction algorithm is a simple means for constructing a maximally-valid interpretation of an information state trajectory in terms of feasible physical state trajectory. We may combine the two procedures to generate physical state sample paths $s(t)$, $t_0 \leq t \leq t_0 + T$, of some specified length $T$. Initially we know that the components of information state must be extended so that their total process time is at least $T$, and so we repeatedly (using the Euler-Maryama approximation) expand the maxflux process (whose identity may shift during expansion) until the total process time reaches $T$. We then perform trajectory retraction on what we have just generated and compute the maximal value of time for the corresponding feasible physical state trajectory, $t_{max}$, which follows trivially from the retracted trajectories. In general $t_{max} < T$, and the information states must be further extended (from their endpoints that correspond to $t_{max}$). This can be done in a variety of ways; for example, by tracing component information state level-crossings to identify the remaining physical-state transitions, or by direct simulation.

#### 6.3.2.2 Normal approximation and interpolation

The preceding numerical SDE approach may be viewed as producing information state component sample-path trajectories through a superposition of many small normal increments, then using flux constraints to truncate the sample-paths.

We now consider an approach that may be viewed as taking one large step, or rather as using the diffusion model to gauge the distribution of a given information state component if we were to follow it out to any specified horizon. The slope defined by linear interpolation to the terminal information state is a (normal) random variable, which in turn appears in the coefficient matrix of a linear system of equations describing flux-constraints (and whose solution determines the *joint* values of information state).

Using the special structure of the flux-constraint matrix we can invert the linear flux-constraint system. Finally, we use a Taylor-series estimation technique to approximate the mean values of joint information state, or equivalently, the expected values of the Markov-chain transition frequencies, which for the case of known rewards and total undiscounted reward criterion determines the value of the Markov reward process.

In an attempt to make this approach more concrete, we present the steps for the simple case of a Markov chain with two physical states. Let the parameters for the (matrix beta) prior for $P$ be

$$M^o \stackrel{\text{def}}{=} \begin{bmatrix} m_{11}^0 & m_{12}^0 \\ m_{21}^0 & m_{22}^0 \end{bmatrix}.$$

Begin by transforming to $(x, \tau)$ coordinates:

$$\begin{bmatrix} \tau_1^0 \\ x_1^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} m_{11}^0 \\ m_{12}^0 \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} \tau_2^0 \\ x_2^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} m_{22}^0 \\ m_{21}^0 \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

Introducing a diffusion approximation for each of these components as in Section 6.3.1 implies that each embedded information state component has a density that is Gaussian with mean

$$E(x_i(\tau_i)) = \frac{x_i(\tau_i^0)}{\tau_i^0 + 2}\tau_i + \frac{2x_i(\tau_i^0)}{\tau_i^0 + 2} \qquad i = 1, 2$$

and second moment

$$E(x_i^2(\tau_i)) = c_i(\tau_i + 2)^2 - (\tau_i + 2) \qquad i = 1, 2$$

where

$$c_i = \frac{x_i^2(\tau_i) + (\tau_i^0 + 2)}{(\tau_i^0 + 2)^2} \qquad i = 1, 2.$$

This implies that the variance of $x$ is a quadratic function of $\tau$.

Suppressing subscripts for the moment, define the slope, $s(\Delta\tau)$, as the slope of linearly-interpolated trajectory of $x$ defined by its values at $\tau_0$ and $\tau_0 + \Delta\tau$ :

$$s(\Delta\tau) \overset{\text{def}}{=} \frac{x(\tau_0 + \Delta\tau) - x(\tau_0)}{\Delta\tau}$$

whose expected value is constant:

$$E(s(\Delta\tau)) = \frac{x(\tau_0)}{\tau_0 + 2}.$$

The variance of the slope is

$$Var(s(\Delta\tau)) = \left(\frac{1}{\Delta\tau}\right)^2 Var(x(\tau_0 + \Delta\tau)).$$

Suppose the Markov chain starts in physical state 1. The flux constraints require that observed state transition frequency counts satisfy

$$f_{12} - f_{21} = \delta_{2v},$$

where $v$ is the final state, together with

$$f_{11} + f_{12} + f_{22} + f_{21} = n,$$

where $n$ is the total number of observed transitions.

In matrix form, the flux constraints are

$$\begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{22} \\ f_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} \quad or \quad \begin{bmatrix} 1 \\ n \end{bmatrix}.$$

(The two right-hand sides correspond to the final state being equal to STATE 1 or STATE 2, respectively).

The transition frequency matrix is just the difference of the prior and posterior matrices of matrix beta parameters. To write the flux constraints in terms of $(x, \tau)$

parameters we first consider the transformation to matrix beta parameters, and then to transition frequencies; *e.g.*, for components associated with state 1,

$$\begin{bmatrix} m_{11} \\ m_{12} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \tau_1 \\ x_1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

which implies that

$$\begin{bmatrix} f_{11} \\ f_{12} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \Delta\tau_1 \\ \Delta x_1 \end{bmatrix}.$$

Thus, in terms of $(x, \tau)$ coordinates, the flux constraints may be written as

$$\begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -1 & & \\ 1 & 1 & & \\ & & 1 & -1 \\ & & 1 & 1 \end{bmatrix}\begin{bmatrix} \Delta\tau_1 \\ \Delta x_1 \\ \Delta\tau_2 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} \quad or \quad \begin{bmatrix} 1 \\ n \end{bmatrix}$$

or

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 0 & 2 & 0 \end{bmatrix}\begin{bmatrix} \Delta\tau_1 \\ \Delta x_1 \\ \Delta\tau_2 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} \quad or \quad \begin{bmatrix} 1 \\ n \end{bmatrix}.$$

Now in time $\Delta\tau_1$, $\Delta x_1 \approx s_1\Delta\tau_1$, and in time $\Delta\tau_2$, $\Delta x_2 \approx s_2\Delta\tau_2$, which make the flux constraints

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \Delta\tau_1 \\ s_1\Delta\tau_1 \\ \Delta\tau_2 \\ s_2\Delta\tau_2 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} \quad or \quad \begin{bmatrix} 2 \\ n \end{bmatrix}$$

or

$$\begin{bmatrix} 1+s_1 & -(1+s_2) \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta\tau_1 \\ \Delta\tau_2 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} \quad or \quad \begin{bmatrix} 2 \\ n \end{bmatrix}.$$

The full system (including equations of the form $\Delta x = s\Delta\tau$) is:

$$\begin{bmatrix} 1+s_1 & -(1+s_2) & & \\ 1 & 1 & & \\ s_1 & & -1 & \\ & s_2 & & -1 \end{bmatrix} \begin{bmatrix} \Delta\tau_1 \\ \Delta\tau_2 \\ \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ n \\ 0 \\ 0 \end{bmatrix} \quad or \quad \begin{bmatrix} 2 \\ n \\ 0 \\ 0 \end{bmatrix}.$$

Call the coefficient matrix $A$. It has the partitioned form

$$A = \begin{bmatrix} P & 0 \\ D & -I \end{bmatrix}$$

whose inverse is

$$\begin{aligned} A^{-1} &= \begin{bmatrix} P^{-1} & 0 \\ DP^{-1} & -I \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{\begin{bmatrix} 1 & s_2+1 \\ -1 & s_1+1 \end{bmatrix}}{s_1+s_2+2} & 0 \\ \dfrac{\begin{bmatrix} s_1 & s_1(s_2+1) \\ -s_2 & s_2(s_1+1) \end{bmatrix}}{s_1+s_2+2} & \begin{bmatrix} -1 & \\ & -1 \end{bmatrix} \end{bmatrix}. \end{aligned}$$

### 6.3.2.3 Taylor-series approximation for mean

We now make use of the fact (see Papoulis, 1991, p. 156, for example) that if a function, $g(x, y)$ is sufficiently smooth near the point $(\mu_x, \mu_y)$, then the mean $\mu_g$ (and variance $\sigma_g^2$) of $g(X, Y)$ can be estimated in terms of the mean, variance, and covariance of $X$ and $Y$:

$$\mu_g \cong g(\mu_x, \mu_y) + \frac{1}{2}\left[\frac{\partial^2 g}{\partial x^2}\sigma_x^2 + 2\frac{\partial^2 g}{\partial x \partial y} r\sigma_x\sigma_y + \frac{\partial^2 g}{\partial y^2}\sigma_y^2\right],$$

where the derivatives of $g$ are evaluated at $x = \mu_x$ and $y = \mu_y$, are $r$ is the correlation coefficient.

In our case, the slopes, $s_1$ and $s_2$, are independent, making $r$ zero:

$$\mu_g \cong g(\overline{s}_1, \overline{s}_2) + \frac{1}{2}\left[\frac{\partial^2 g}{\partial s_1^2}\sigma_{s_1}^2 + \frac{\partial^2 g}{\partial s_2^2}\sigma_{s_2}^2\right],$$

Estimates for the mean value of any desired element of $A^{-1}$ can be derived by applying the Taylor-series formula in which the function $g$ assumes the form of the corresponding element's functional form. For example, the second column of $A^{-1}$ is:

$$\begin{bmatrix} \frac{s_2+1}{s_1+s_2+2} \\ \frac{s_1+1}{s_1+s_2+2} \\ \frac{s_1(s_2+1)}{s_1+s_2+2} \\ \frac{s_2(s_1+1)}{s_1+s_2+2} \end{bmatrix}$$

and the Taylor-series estimate for the mean of the third element, for example, in this column is given by

$$\begin{aligned} \mu\left(\frac{s_1(s_2+1)}{s_1+s_2+2}\right) &\cong \frac{\overline{s}_1(\overline{s}_2+1)}{\overline{s}_1+\overline{s}_2+2} \\ &+ \left[\frac{\overline{s}_1(\overline{s}_2+1)}{(\overline{s}_1+\overline{s}_2+2)^3} - \frac{\overline{s}_2+1}{(\overline{s}_1+\overline{s}_2+2)^2}\right]\sigma_{s_1}^2 - \frac{\overline{s}_1(\overline{s}_1+1)}{(\overline{s}_1+\overline{s}_2+2)^3}\sigma_{s_2}^2. \end{aligned}$$

Finally, we may obtain an estimate for joint transition frequencies by substituting values for the slope means, $\overline{s}_i$, and variances, $\sigma_{s_i}^2$, derived previously. Estimates for the mean of the joint distribution over $x_i$ and $\tau_i$ can be derived by multiplying the estimate for the mean of $A^{-1}$ by the appropriate right-hand side, yielding an estimate for mean $x_i$ and $\tau_i$. A simple linear transformation translates the result back into the original coordinate system that has an interpretation in terms of transition frequencies.

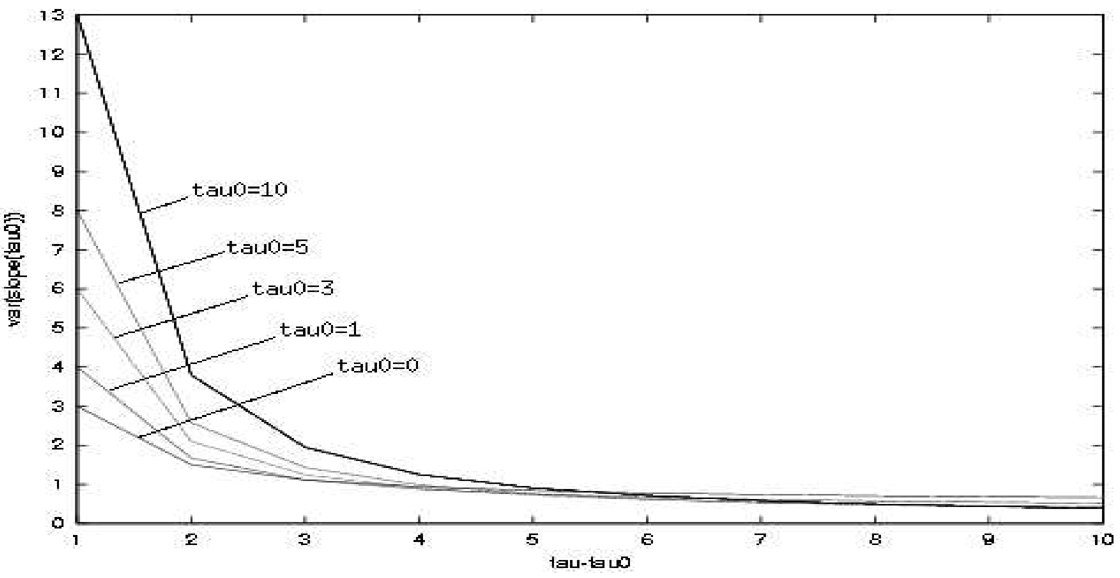


Figure 6.3: Plots of slope variance, $\sigma_s^2$, as a function of time, for various values of $\tau_0$.

### 6.3.3 Some details

#### 6.3.3.1 Slope Variance

It can be shown that

$$Var\,(x(\tau)) = \frac{(\tau+2)(\tau+1)}{\tau_0+2},$$

and hence that

$$\begin{aligned}\sigma_s^2 &= \tfrac{1}{(\tau-\tau_0)^2} Var(x(\tau)) \\ &= \tfrac{(\tau+2)(\tau+1)}{(\tau-\tau_0)^2(\tau_0+2)},\end{aligned}$$

whose asymptotic value is $\frac{1}{\tau_0+2}$.

In Figure 6.3, we plot $\sigma_s^2$ for various values of $\tau_0$.

For computing estimates for joint transition frequencies, we suggest two possibilities: (1) Use the asymptotic values for slope variances, or (2) Begin by using the asymptotic slope variance values in solving the linear system for mean $x$ and $\tau$, then substitute the mean $\tau$ into the formula for $\sigma_s^2$ and recompute the solution to the linear system.

#### 6.3.3.2 Flux constraints

Previously, we stated that the flux constraints require (assuming STATE 1 is the starting state) that the transition frequency counts satisfy

$$f_{12} - f_{21} = \delta_{2v},$$

where $v$ is the final state, together with

$$f_{11} + f_{12} + f_{22} + f_{21} = n,$$

where $n$ is the total number of observed transitions.

Adopting the linearly-interpolated model for increments in information state components implies that $f_{12}$ and $f_{21}$ are linear functions of $f_{11}$ and $f_{22}$, respectively:

$$\begin{aligned} f_{12} &= s_1 f_{11} \\ f_{21} &= s_2 f_{22}, \end{aligned}$$

where the $s_i$ now denote slopes in the transition frequency planes, and we can rewrite the second flux constraint as:

$$\frac{s_1 + 1}{s_1} f_{12} + \frac{s_2 + 1}{s_2} f_{21} = n.$$

In Figure 6.4, we sketch the geometric relationships implied by the flux constraints. The first flux constraint defines two hyperplanes, each corresponding to a particular terminal state. The second flux constraint defines a third hyperplane whose intersection with the previously defined hyperplanes determines transition frequency counts that would satisfy the flux constraint equations.

In general, the solution to the flux constraint system will be a set of transition frequencies that are real numbers rather than integers. True interpolated information-state increment component trajectories would be "stair-step" functions, of the form

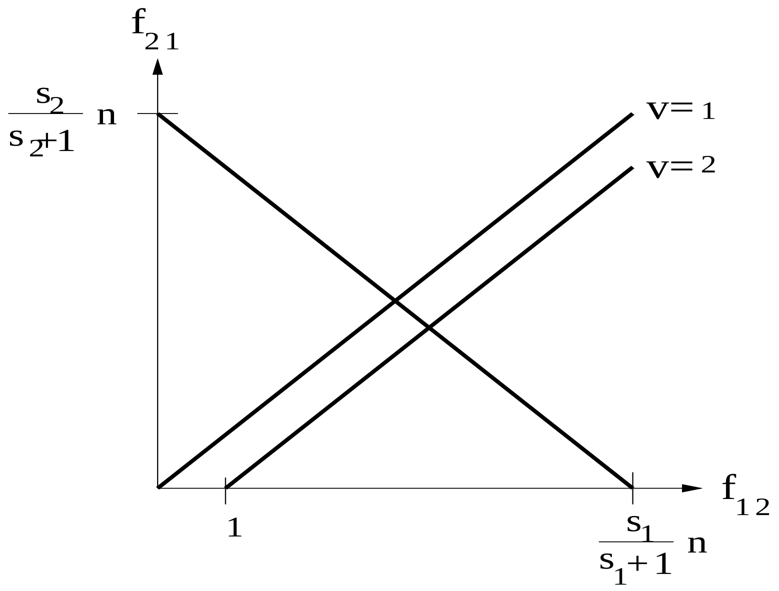


Figure 6.4: Flux constraint geometry.

$f_{12} = \lfloor s_1 f_{11} \rfloor$ for example. Allowing transition frequencies to take on real values facilitates analysis, but introduces another source of approximation error.

It may be convenient to adopt the convention of assuming that the right-hand side of the first (flux-balance) constraint is zero. This implies that we are presuming a sort of conservation of transitional flux across the boundaries of each state, and that we ignore deviations from this presumption posed by a detailed consideration of terminal state identity. For our ongoing example, this implies that we choose the hyperplane corresponding to STATE 1 being the terminal state, and the transition frequencies are defined by the intersection of this hyperplane with the hyperplane defined by there being a specified number of total transitions. Again, this assumption introduces another source of approximation error, but facilitates analysis. Note that, under this assumption, we need only compute one column of the coefficient matrix inverse, $A^{-1}$.

### 6.3.4 Summary and Example

#### 6.3.4.1 Summary

Here, then, are the details of an analytic procedure for approximating the value of a Bayes-adaptive Markov reward process with two physical states:

**Step 1.** Translate from the matrix-beta parameters, which specify prior uncertainty in the unknown transition probabilities, to $(x, \tau)$ coordinates:

$$\begin{bmatrix} \tau_1^0 \\ x_1^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} m_{11}^0 \\ m_{12}^0 \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} \tau_2^0 \\ x_2^0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} m_{22}^0 \\ m_{21}^0 \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix}.$$

**Step 2.** An approximation for the expected increments in the values of the information state trajectories, in $(x, \tau)$ coordinates, is given by the expected value of the second column of the inverse of the flux constraint matrix, $A^{-1}$, multiplied by the total number of transitions, $n$. Utilizing the Taylor-series approximation, the expected values of the elements of the second column of $A^{-1}$ are given by:

$$\begin{aligned}
\mu\left(\tfrac{s_2+1}{s_1+s_2+2}\right) &\cong \tfrac{\bar{s}_2+1}{\bar{s}_1+\bar{s}_2+2} \\
&\quad + \tfrac{\bar{s}_2+1}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_1}^2 - \tfrac{\bar{s}_1+1}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_2}^2 \\
\mu\left(\tfrac{s_1+1}{s_1+s_2+2}\right) &\cong \tfrac{\bar{s}_1+1}{\bar{s}_1+\bar{s}_2+2} \\
&\quad + \tfrac{-(\bar{s}_2+1)}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_1}^2 + \tfrac{\bar{s}_1+1}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_2}^2 \\
\mu\left(\tfrac{s_1(s_2+1)}{s_1+s_2+2}\right) &\cong \tfrac{\bar{s}_1(\bar{s}_2+1)}{\bar{s}_1+\bar{s}_2+2} \\
&\quad + \left[\tfrac{-(\bar{s}_2+1)}{(\bar{s}_1+\bar{s}_2+2)^2} + \tfrac{\bar{s}_1(\bar{s}_2+1)}{(\bar{s}_1+\bar{s}_2+2)^3}\right]\sigma_{s_1}^2 \\
&\quad - \tfrac{\bar{s}_1(\bar{s}_1+1)}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_2}^2 \\
\mu\left(\tfrac{s_2(s_1+1)}{s_1+s_2+2}\right) &\cong \tfrac{\bar{s}_2(\bar{s}_1+1)}{\bar{s}_1+\bar{s}_2+2} \\
&\quad - \tfrac{\bar{s}_2(\bar{s}_2+1)}{(\bar{s}_1+\bar{s}_2+2)^3}\sigma_{s_1}^2 \\
&\quad + \left[\tfrac{-(\bar{s}_1+1)}{(\bar{s}_1+\bar{s}_2+2)^2} + \tfrac{\bar{s}_2(\bar{s}_1+1)}{(\bar{s}_1+\bar{s}_2+2)^3}\right]\sigma_{s_2}^2,
\end{aligned}$$

where

$$\bar{s}_i = E(s_i) = \frac{x_i(\tau_i^0)}{\tau_i^0 + 2}, \quad i = 1, 2,$$

and the slope variances, $\sigma_{s_i}^2$ , $i = 1, 2$, may be defined by their asymptotic values,

$$\sigma_{s_i}^2 = \frac{1}{\tau_i^0 + 2},$$

or through an iterative procedure of the sort described previously in Section 3.5.1.

**Step 3.** Multiply the approximate $\mu(\cdot)$-values from Step 2 by $n$, the total number of transitions, to get approximate values for the expected increments in $(x, \tau)$ coordinates. Perform a simple linear transformation of the result to get an approximation for the expected increments in matrix-beta parameters, which are the expected transition frequencies:

$$E \begin{bmatrix} f_{11} \\ f_{12} \\ f_{22} \\ f_{21} \end{bmatrix} \cong \frac{1}{2} \begin{bmatrix} 1 & -1 & & \\ 1 & 1 & & \\ & & 1 & -1 \\ & & 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta\tau_1 \\ \Delta x_1 \\ \Delta\tau_2 \\ \Delta x_2 \end{bmatrix}.$$

Finally, one may take the inner product of the approximate expected transition frequencies with the corresponding expected immediate rewards to form an approximation for the value of the Bayes-adaptive Markov reward process.

#### 6.3.4.2 Example (2 physical states, horizon=25)

As a simple test of our analytic approach we consider a two-state Markov chain whose transition probability uncertainty is characterized by a matrix-beta prior with parameters:

$$M^o \equiv \begin{bmatrix} m_{11}^0 & m_{12}^0 \\ m_{21}^0 & m_{22}^0 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}.$$

Over the course of 25 state transitions (starting from STATE 1) a Monte-Carlo estimate, which we may regard as being exact, for the mean transition frequencies is:

$$\overline{f}_{MC} = \begin{bmatrix} 3.04 & 4.49 \\ 3.75 & 13.7 \end{bmatrix}$$

while our analytic approximation technique leads to:

$$\overline{f}_{Anal_1} = \begin{bmatrix} 2.67 & 3.48 \\ 3.48 & 15.4 \end{bmatrix}.$$

We note that the computational complexity of our analytic approach is independent of the time horizon. In fact, we might expect that its estimates would improve in quality as the time-horizon increases, since it adopts asymptotic values for slope variances and since the relative error of assuming flux balance and ignoring the details of terminal state identity decreases as the number of total transitions increases. Our approximation technique requires coordinate transformations entailing work of order $O(N^2)$, where $N$ denotes the number of physical states, and evaluation of the $A^{-1}$ column formulae, which requires $O(N)$ computation.

If we adopt a right-hand-side of the flux constraint equations that is consistent with the terminal state being STATE 2, then two columns of $E(A^{-1})$ enter into our approximation formulae. (The approximate expected increment in $(x, \tau)$ coordinates is given by our previous estimate plus $2\times$ first column of $E(A^{-1})$ .) This leads to an estimate equal to

$$\overline{f}_{Anal_2} = \begin{bmatrix} 3.19 & 4.23 \\ 3.23 & 14.3 \end{bmatrix}.$$

A reasonable approach would be to average the estimates in some way. If we form the certainty-equivalent estimate for the transition probability matrix from its prior matrix-beta parameters, $\overline{P}_{CE} \equiv \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{4} & \frac{3}{4} \end{bmatrix}$, then solve for the associated stationary distribution, $\pi$, via $\pi = \pi P$, $\sum \pi_i = 1$, we may construct a $\pi-$weighted combination of the $\overline{f}_{Anal_i}$ (for our example, $\pi = \begin{bmatrix} \frac{3}{11} & \frac{8}{11} \end{bmatrix}$):

$$\overline{f}_{Anal_\pi} = \begin{bmatrix} 3.05 & 4.02 \\ 3.30 & 14.6 \end{bmatrix}.$$

Computing the stationary distribution, $\pi$, requires inverting a matrix of dimension $N$. Direct methods are $O(N^3)$. An approximate iterative method suggested by the formula $\pi^{(n)} = \pi^{(0)}P^n$ is order $O(N^2)$ per iteration, with $\pi^{(n)} \rightarrow \pi$ geometrically at a rate that depends upon the eigenvalues of $P$. We may therefore incorporate some aspect of terminal state identity into our approximation analysis while retaining an $O(N^2)$ bound on the total work that we do. We note that, in theory, it is possible to compute $E_P(\pi)$ with respect to some specified prior for $P$ (Martin, 1967, p. 73); however the successive approximation scheme proposed by Martin, in effect, requires the enumeration of the set of reachable hyperstates (though for the $2 \times 2$ case, Martin derives a complicated series expansion formula when $P$ has a matrix beta distribution).

## 6.4 Extending the analytic approach to decision processes

### 6.4.1 BAMDP's and stochastic policies

We now consider certain issues of control—the construction of optimal policies for Markov decision processes with unknown transition probabilities. Transition probabilities are now indexed by action (which we assume to be drawn from some finite set); we write $p_{ij}^a$ for the probability of transitioning from physical state $i$ to physical state $j$ if we apply action $a$.

We refer to the resulting model for Markov decision process dynamics for a chain with uncertainty in its transition probabilities modeled in a Bayesian way as a *Bayes-Adaptive Markov Decision Process* (BAMDP). One can generalize our approach of previous sections to this generalized process.

For example, the matrix of matrix beta parameters, $M$, which is simply related to transition frequencies and to information state components, $x$, can now be indexed by action as well. In what follows, the notational convention will be to use subscripts to denote physical-state components and superscripts to denote action components. For example, $f_{ij}^a$ denotes the number of observed transitions from physcial state $i$ to physical state $j$ when action $a$ has been applied. Information state, $x$, can be taken to be affine-transformed versions of the entire collection of $m_{ij}^a$ 's, which serve as parameters that define associated beta distributions describing our uncertainty in the $p_{ij}^a$'s.

We shall be considering *stochastic* policies, which in general map each hyperstate to a probability distribution over actions. We shall use $\pi$ to denote the policy, and subscripts, superscripts, and arguments will indicate specific components (Note that this differs from our previous use of $\pi$ to denote the stationary distribution of a Markov chain). For example, $\pi_s^a(x)$ denotes the probability of taking action $a$ given physical state $s$ and information state $x$; for the case of two physical states and two actions, the policy may be specified by the four components: $\pi_1^1(x)$, $\pi_1^2(x)$, $\pi_2^1(x)$, and $\pi_2^2(x)$ (two of these components suffice).

### 6.4.2 Information drift

Now consider the evolution of the various components, $x_s^a$, of information state. As for the case of information state dynamics associated with Markov chains, each of these components moves about upon identical terrain, the only differences are due to differing priors and differing rates of transition—each component has a local process clock that ticks only when we are in the component's corresponding physical state, *take the action indicated by the component's superscript*, and observe a transition. Note that component transition rates now depend upon policy as well as physical state. The policy modulates how the action components of information state are to be combined in order to construct a mixture process whose level-crossings may be interpreted as changes in physical state.

Let $t$ denote the global process time variable for the BAMDP, let $\tau_s$ denote the process time variable for the physical state $s$ component, and let $\tau_s^a$ denote the physical state $s$ action $a$ time component. Component process transition rates are defined by:

$$d\tau_s^a \;=\; \pi_s^a(x) d\tau_s,$$

and each information state component evolves upon identical terrain:

$$dx_s^a \;=\; \tfrac{x_s^a}{\tau_s^a+2} d\tau_s^a + dW_{\tau_s^a},$$

which can be rewritten in terms of the physical state process time variable increment:

$$dx_s^a \;=\; \tfrac{x_s^a{}_s^a\pi_s^a}{\tau_s^a+2} d\tau_s + \sqrt{\pi_s^a} dW_{\tau_s} \;\cdot$$

Mixture processes are defined by their components and the stochastic policy:

$$\begin{aligned} dx_s &= \textstyle\sum_a dx_s^a \\ &= \textstyle\sum_a \left( \tfrac{x_s^a \pi_s^a}{\tau_s^a+2} d\tau_s + \sqrt{\pi_s^a} dW_{\tau_s} \right). \end{aligned}$$

### 6.4.3 Example (2 physical states, 2 actions)

Local process clock variables:

$$\begin{aligned}
d\tau_1^1 &= \pi_1^1(x)d\tau_1 \\
d\tau_1^2 &= \pi_1^2(x)d\tau_1 \\
d\tau_2^1 &= \pi_2^1(x)d\tau_2 \\
d\tau_2^2 &= \pi_2^2(x)d\tau_2.
\end{aligned}$$

Each information component evolves at these differing rates upon identical terrain:

$$\begin{aligned}
dx_1^1 &= \tfrac{x_1^1}{\tau_1^1+2}d\tau_1^1 + dW_{\tau_1^1} \\
dx_1^2 &= \tfrac{x_1^2}{\tau_1^2+2}d\tau_1^2 + dW_{\tau_1^2} \\
dx_2^1 &= \tfrac{x_2^1}{\tau_2^1+2}d\tau_2^1 + dW_{\tau_2^1} \\
dx_2^2 &= \tfrac{x_2^2}{\tau_2^2+2}d\tau_2^2 + dW_{\tau_2^2},
\end{aligned}$$

and the mixture processes are defined by their components and the stochastic policy:

$$\begin{aligned}
dx_1 &= dx_1^1 + dx_1^2 \\
&= \left[\tfrac{x_1^1\pi_1^1}{\tau_1^1+2} + \tfrac{x_1^2\pi_1^2}{\tau_1^2+2}\right] d\tau_1 + \left(\sqrt{\pi_1^1} + \sqrt{\pi_1^2}\right) dW_{\tau_1} \\
dx_2 &= dx_2^1 + dx_2^2 \\
&= \left[\tfrac{x_2^1\pi_2^1}{\tau_2^1+2} + \tfrac{x_2^2\pi_2^2}{\tau_2^2+2}\right] d\tau_2 + \left(\sqrt{\pi_2^1} + \sqrt{\pi_2^2}\right) dW_{\tau_2}.
\end{aligned}$$

#### 6.4.3.1 Atomic construction of sample paths

Conceptually, we could construct valid information state sample paths by following information component diffusions, using the policy to construct the mixture process, and by following the mixture process until it level-crosses. Suppose we start in physical state 1. Then, $d\tau_1 = dt$, while $d\tau_2 = 0$. This implies that $d\tau_1^1$ and $d\tau_1^2$ are $\pi_1^1(x)$ and $\pi_1^2(x)$ respectively, while $d\tau_2^1$ and $d\tau_2^2$ are both zero. We then integrate the $dx_1^1$ and $dx_1^2$ equations forward, constructing the mixture process $x_1$ from the sum of the component process increments. When $x_1$ crosses a level corresponding to an integral value of $m_{12}$, this event indicates a transition to physical state 2, at which point $d\tau_2 = dt$ and the information components corresponding to physical state 2 become active. We construct $x_2$ from its components and follow its trajectory until it crosses, and so on. Notice that the transition rates of the component processes are governed by the policy, which, in general, depends upon both the active and inactive information state components. Constructing sample paths in this way could serve as a basis for a numerical, Monte-Carlo approach for estimating mean transition frequencies and (for known rewards and total reward value-criterion) the value of a Markov decision process governed by a stochastic policy.

#### 6.4.3.2 Analytic approximation

We now wish to develop an analytic method for approximating mean transition frequencies that avoids the fine-grained detail of the previous method. Ideally, we would like to proceed as we did with Markov chains.

Suppose we have a collection of information component sample paths, $x_s^a$. The stochastic policy $\pi$ modulates how these components are combined to form the mixture processes $x_s$, which must obey constraints of flux as for the Markov chain case.

To complicate matters, note that $\pi$ depends on $x$—on *all* components $x_s^a$. Consider our running example and the components $\pi_1^1$ and $\pi_1^2$ that mix the components $x_1^1$ and $x_1^2$ together. $\pi_1^1$ and $\pi_1^2$ both depend upon all components: $x_1^1$ and $x_1^2$ *and* $x_2^1$ and $x_2^2$. However, for some given value of $\tau_1$ we do not know the corresponding $\tau_2$ (or $\tau_2^1$ or $\tau_2^2$)—we do not know how far out to go along the $x_2^1$ and $x_2^2$ trajectories.

#### 6.4.3.3 Stationary stochastic policies

Suppose $\pi_1^1$ and $\pi_1^2$ were constant for the time interval of interest; *i.e.*, suppose that $\pi_s^a(\cdot)$ are constant as functions of $x$. Then the mixing process is well-defined, and we are led to a numerical approach for generating sample paths corresponding to stationary stochastic policies. For the two-state / two-action case, for instance, we could integrate the $x_1^1$ and $x_1^2$ and $x_2^1$ and $x_2^2$ processes forward, then combine these components together using the stationary stochastic policy, forming the mixture processes $x_1$ and $x_2$. Finally, we could retract the mixture trajectories so that they satisfy flux constraints and correspond to a valid physical state trajectory.

Analytically, we may construct the distribution of each information state component, $x_s$, as the mixture (weighed by $\pi_s^a$) of the Gaussian sub-components, $x_s^a$. Conceptually, we may consider linearly-interpolated trajectories to the resulting Gaussian process. We can then incorporate the effect of flux constraints to model the joint behavior of information state components just as we did for Markov chains, the only difference being that the distributions modeling information state components, $x_s$, are defined by the mixture of their sub-components, $x_s^a$.

#### 6.4.3.4 Example (2 states / 2 actions)

We are given the matrix beta prior for the generalized transition probability matrix (transition probabilities for all actions); that is, we are given values for prior parameters for the matrix beta parameter matrix:

$$ {}^0M \equiv \begin{bmatrix} {}^0m_{11}^1 & {}^0m_{12}^1 \\ {}^0m_{21}^1 & {}^0m_{22}^1 \\ {}^0m_{11}^2 & {}^0m_{12}^2 \\ {}^0m_{21}^2 & {}^0m_{22}^2 \end{bmatrix}. $$

We begin by translating to $(x, \tau)$-coordinates:

$$ \begin{bmatrix} {}^0\tau_1^k \\ {}^0x_1^k \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} {}^0m_{11}^k \\ {}^0m_{12}^k \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix} \qquad k = 1, 2 $$

and

$$ \begin{bmatrix} {}^0\tau_2^k \\ {}^0x_2^k \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} {}^0m_{22}^k \\ {}^0m_{22}^k \end{bmatrix} + \begin{bmatrix} -2 \\ 0 \end{bmatrix} \qquad k = 1, 2. $$

Each information state component evolves via:

$$dx_s^a \;=\; \tfrac{x_s^a}{\tau_s^a+2} d\tau_s^a + dW_{\tau_s^a} \qquad x_s^a({}^0\tau_s^a) = \;{}^0x_s^a,$$

a Gaussian process, with mean and variance derived exactly as before for the Markov chain case. Now consider the linearly-interpolated trajectory to the mixture process, $\pi_s^1 x_s^1 + \pi_s^2 x_s^2$. Its expected value is

$$\begin{aligned} E\left(\textstyle\sum_a \pi_s^a x_s^a(\tau)\right) &= \textstyle\sum_a \pi_s^a E\left(x_s^a(\tau)\right) \\ &= \textstyle\sum_a \pi_s^a x_s^a({}^0\tau_s^a) \tfrac{\tau+2}{{}^0\tau_s^a+2}. \end{aligned}$$

The variance of the linearly-interpolated trajectory to the mixture process is given by

$$\begin{aligned} Var\left(\textstyle\sum_a \pi_s^a x_s^a(\tau)\right) &= \textstyle\sum_a \left(\pi_s^a\right)^2 Var\left(x_s^a(\tau)\right) \\ &= \textstyle\sum_a \left(\pi_s^a\right)^2 \tfrac{(\tau+2)(\tau+1)}{{}^0\tau_s^a+2}. \end{aligned}$$

The mean slope of linearly-interpolated trajectories is

$$E(slope_s) = \sum_a \pi_s^a \frac{x_s^a({}^0\tau_s^a)}{{}^0\tau_s^a + 2},$$

while an asymptotic measure of its variance is given by

$$Var(slope_s) \to \sum_a \frac{\left(\pi_s^a\right)^2}{{}^0\tau_s^a + 2}.$$

An estimate for the mean transition frequencies associated with a stationary stochastic policy may be obtained by using these means and variances in the formulae provided previously (in Step 2) of Section 6.4.3.1.

#### 6.4.3.5 Numerical Example

Consider the case of two physical states and two admissible actions for each state. Suppose the matrix beta prior for the generalized transition matrix is

$$^{0}M \equiv \begin{bmatrix} ^{0}m_{11}^{1} & ^{0}m_{12}^{1} \\ ^{0}m_{21}^{1} & ^{0}m_{22}^{1} \\ ^{0}m_{11}^{2} & ^{0}m_{12}^{2} \\ ^{0}m_{21}^{2} & ^{0}m_{22}^{2} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 3 & 1 \\ 1 & 9 \end{bmatrix},$$

and that the stationary stochastic policy is specified by

$$\begin{bmatrix} \pi_1^1 & \pi_1^2 \\ \pi_2^1 & \pi_2^2 \end{bmatrix} = \begin{bmatrix} .8 & .2 \\ .2 & .8 \end{bmatrix}.$$

Our analytic estimation procedure leads to mean transition frequency estimates:

$$F_{Anal_\pi} = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix} = \begin{bmatrix} 2.79 & 2.84 \\ 2.02 & 17.35 \end{bmatrix}$$

(the weighting factor here is the steady-state distribution associated with the transition probability matrix formed from the mixture of certainty-equivalent "pure" policies) which may be compared with mean estimates obtained by Monte-Carlo simulation:

$$F_{MonteCarlo} = \begin{bmatrix} 2.90 & 3.12 \\ 2.31 & 16.66 \end{bmatrix}.$$

## 6.5 Summary

Our goal has been the synthesis of an analytic procedure for estimating the mean state-transition frequencies of a Markov chain whose transition probabilities are unknown but modeled in a Bayesian way. We have noted that information state components (the parameters specifying distributions that model uncertainty in transition probabilities leading from each state) may be viewed as embedded Markov chains that are amenable to diffusion modeling (see Figure 6.5). Analysis yields simple formulae for means and variances of Gaussian processes that model the flow of information state density. Conceptually we may imagine generating sample paths of joint information state by: (1) Generating a set of terminal information states consistent with the Gaussian process models, (2) Constructing a corresponding set of linearly-interpolated information-state trajectories, and (3) Retracting these trajectories so that they obey flux constraints, so that together they form a set of trajectories whose joint extent is consistent with a feasible physical-state trajectory. Our analysis gauges the mean result of performing these sample path constructions many times and computing the mean terminal joint information state. Our approach relies upon analytic methods, rather than simulation or brute-force computation, and its complexity scales polynomially with the number of physical states ($O(N^2)$), rather than exponentially with the time horizon.

If we add a known reward structure to our Bayes-adaptive Markov chain, then an estimate for the (finite-horizon, total, undiscounted) value of the resulting reward process follows from taking the inner product of mean state-transition frequencies with expected immediate rewards. In the previous section, we showed how our approach can provide estimates for the value of a stationary, stochastic policy. The obvious next step is to couple the policy evaluation scheme proposed here with a procedure for policy improvement, and one obvious approach would involve stepping along the (estimated) gradient of value with respect to policy component parameters, $\pi_s^a$. One step toward more general, and presumably more effective control, would be to adopt *piecewise* stationary stochastic policies. For example, one could derive an approximately-optimal stationary stochastic policy over the entire time interval (computed, for instance, via gradient ascent), apply the policy over some initial sub-interval, then re-derive an approximately-optimal stationary stochastic policy over

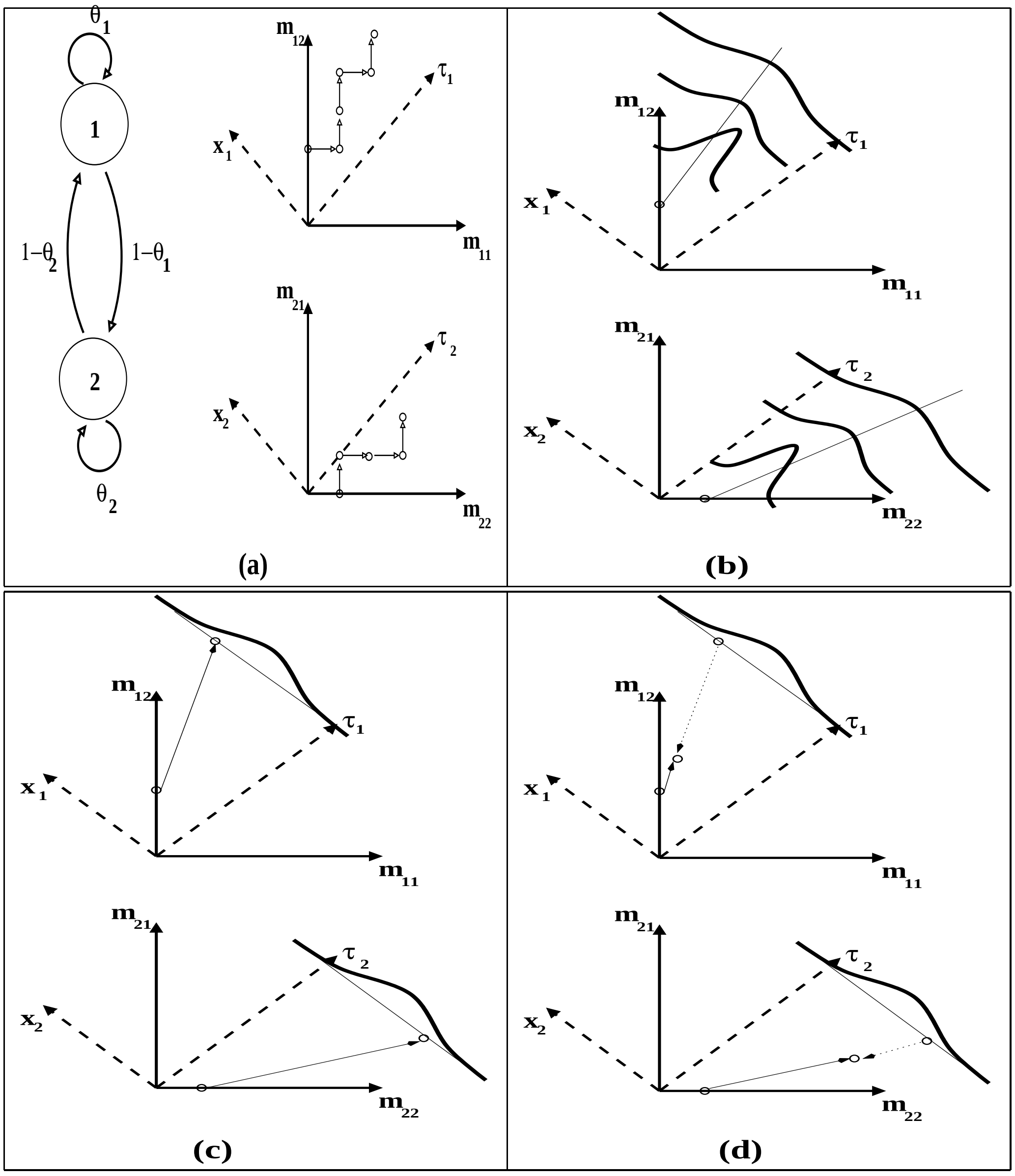


Figure 6.5: A pictorial summary of our approach: (a) Figure 6.2 repeated: information components as embedded Markov chains, (b) Information density modeled as a Gaussian process, (c) Sampling from the Gaussian process and constructing a linearly-interpolated trajectory, (d) Retraction to extents that satisfy flux constraints.

the remaining interval, using the terminal hyperstate at the initial sub-interval as the new initial hyperstate. One may envision an extreme version of this procedure in which we construct a new stationary policy after each state transition.

These ideas are rather preliminary in nature. Some open issues include:

- We conjecture that the basic analytical approach can be generalized to $N$ states.
- We seek a more-refined diffusion model. The variance of the Gaussian model overestimates that of true information state components (though this is counterbalanced by the use of the asymptotic measure of slope variance).
- Our method of combining estimates by weighting them by the stationary distribution associated with the certainty-equivalent transition probability matrix seems sensible, but there may well be a more elegant way of accounting for terminal physical state.
- This chapter has developed an analytical procedure for approximating the value of a BAMDP governed by a fixed (stochastic) policy. We would like to consider how this procedure could serve as the basis for hill-climbing in policy parameter space.

# CHAPTER 7

# CONCLUSION

## 7.1 Summary

Optimal learning is a pattern of behavior that yields the highest expected total reward over the entire duration of an agent's interaction with its environment. The problem of determining an optimal learning strategy is a sort of meta-problem, with optimality defined with respect to a distribution of environments that the agent is likely to encounter. Given this prior uncertainty over possible environments, the optimal-learning agent must collect and use information in an intelligent way, balancing greedy exploitation of certainty-equivalent world models with exploratory actions aimed at discerning the true state of nature.

In this dissertation, we have adopted a Bayesian framework for modeling problems of optimal learning. The Bayesian approach leads to an associated Markov decision process defined over a set of process hyperstates. The number of hyperstates grows exponentially with the planning horizon, rendering conventional dynamic programming algorithms intractable for all but trivial problems. Techniques from the field of reinforcement learning theory have been developed for addressing exactly these kinds of problems, that is, MDP's with extremely-large numbers of states. By utilizing techniques from reinforcement learning theory—specifically the use of parameterized function approximators for representing value functions and policies, and Monte-Carlo sampling of process trajectories—our goal has been the construction of tractable computational procedures that produce good approximations to optimal learning policies.

This dissertation began by presenting some motivating examples of optimal learning problems, and the Bayesian framework for modeling the role of information. We included a survey of reinforcement learning theory, and a mathematical formulation of Markov decision problems with uncertainty in their transition probabilities modeled in a Bayesian way—Bayes-adaptive Markov decision processes (BAMDP's). We presented the details of a classical dynamic programming approach for computing optimal learning policies, and acknowledged its limitations. Many heuristics have been proposed by researchers for balancing exploration with exploitation, and Chapter 2 provided a survey of some of these approaches.

The third chapter considered bandit problems, a special class of BAMDP's for which the sole component of hyperstate is the information state, which models uncertainty in unknown parameters. We regarded bandit problems in two ways. First, we developed a model-free reinforcement learning based algorithm for computing optimal strategies defined in terms of Gittins indices. Second, we showed how localized semi-Markov bandit models could acknowledge some aspect of information gain for general BAMDP's, and how learning strategies could be defined in terms of these models' associated Gittins indices.

Chapter 4 presented two algorithms for approximating optimal learning policies for BAMDP's. Both algorithms refer directly to the full-blown BAMDP version of Bellman's optimality equation defined over hyperstates, but use parameterized function-approximators and Monte-Carlo sampling of process trajectories to preserve computational tractability. The first algorithm is a policy-gradient scheme utilizing gradient ascent in the space of (stochastic) controller parameters. The second algorithm is a relatively straightforward application of reinforcement learning theory's SARSA($\lambda$) algorithm with linear value-function approximators. We showed how the policy-gradient algorithm can be applied to a problem of designing an optimal probe for efficiently identifying unknown transition probabilities of an MDP using online experience.

The fifth chapter began by drawing parallels between BAMDP's and partially-observable Markov decision processes (POMDP's), and showed how certain classical analytical results concerning POMDP's can be generalized to the BAMDP case. Motivated by this correspondence, we then adopted finite-state stochastic automata as BAMDP controllers and derived formulae for the value function and its gradient with respect to controller parameters. These formula provided the basis for Monte-Carlo gradient estimation and policy improvement algorithms that are similar in spirit to Chapter 4's policy-gradient approach.

Chapter 6 described our preliminary efforts aimed at making headway on an analytical, rather than computational, front. We considered the problem of determining the value associated with a BAMDP governed by a fixed policy. For a finite-horizon, expected total-reward value criterion, the problem of determining value becomes one of determining the mean state-transition frequencies. Our analytical approach modeled information state component trajectories by diffusion processes, and acknowledged component interdependencies by means of a linear system of flux-constraints.

## 7.2 Enhanced function approximation and variance reduction

The function approximation architectures we have employed to represent value functions and policies are quite simple, even impoverished, and their lack of representational power may lead to attenuated performance. One reason we have chosen architectures in which parameters appear linearly is that theoretical results exist (Konda & Tsitsilis, 1999; Sutton et al., 1999) that ensure convergence to local optima. Other, more expressive, function approximator architectures may be employed, but their application comes at the expense of limited theoretical guarantees. Perhaps the most critical aspect of function approximator design is the choice of features. We have simply adopted the set of raw sufficient statistics defining the multivariate beta (Dirichlet) distribution as our feature set, but by expanding this set to include, for example, bilinear or higher-order combinations of terms (function approximators remain linear in their parameters), we may boost the power of our approximators and improve performance.

In Chapter 2, we reviewed the work of Satia (1968) in which upper- and lower-bounds on uncertain future return for each node in the hyperstate decision tree allow partial enumeration and termination of some branches of the tree. This idea of using bounds to prune search is closely related to "action-elimination" methods (MacQueen, 1967) of conventional dynamic programming. Considerations of this kind suggest that bounds similar to those derived by Satia, or related approximate bounds on the value of perfect information (Dearden *et al.*, 1998), could be used to bias the Monte-Carlo sampling of hyperstate process trajectories, invoking a form of importance sampling that would reduce estimator variance and speed convergence to approximately-optimal policies.

Importance sampling is just one of many classical variance reduction techniques. Another is the use of common random numbers.

Suppose we have two policies, $\pi_1$ and $\pi_2$, whose performance we wish to compare using simulation. That is, we wish to use simulation to estimate

$$\theta = \theta_1 - \theta_2,$$

where $\theta_1 = E\left(V_{\pi_1}(\overrightarrow{u})\right)$ and $\theta_2 = E\left(V_{\pi_2}(\overrightarrow{u})\right)$, and $\overrightarrow{u}$ is a (potentially-large) vector of random numbers that simulation utilizes in the process of generating random variables (*e.g.*, state transitions or stochastic action-selections), which in turn give rise to particular realizations of controlled process trajectories. Note that if $\vec{u}$ is specified, then $V_\pi(\vec{u})$ is a deterministic quantity. We may think of $\vec{u}$ as specifying a representative pool of "scenarios" that may be used, and possibly re-used, to gauge the efficacy of various policies (for example, by forming the sample mean value of some number of controlled process trajectories).

If we generate a particular vector $\vec{u}_1$ and use it to compute $V_{\pi_1}(\vec{u}_1)$, then should we use the same $\vec{u}_1$ to compute $V_{\pi_2}(\vec{u}_1)$? Or do we get a better estimate of $\theta$ by generating an independent set, $\vec{u}_2$, and using it to compute $V_{\pi_2}(\vec{u}_2)$?

Consider the variance of the estimator $V_{\pi_1}(\vec{u}_1) - V_{\pi_2}(\vec{u}_2)$ of $\theta$ :

$$\mathrm{Var}\,(V_{\pi_1}(\vec{u}_1) - V_{\pi_2}(\vec{u}_2)) = \mathrm{Var}\,(V_{\pi_1}(\vec{u}_1)) + \mathrm{Var}\,(V_{\pi_2}(\vec{u}_2)) - 2\mathrm{Cov}\,(V_{\pi_1}(\vec{u}_1), V_{\pi_2}(\vec{u}_2))\,.$$

We see that if $V_{\pi_1}(\vec{u})$ and $V_{\pi_2}(\vec{u})$ are positively correlated, then the variance of the estimator of $\theta$ will be smaller if we use the same set of random numbers $\vec{u}_1$ to compute both $V_{\pi_1}(\vec{u}_1)$ and $V_{\pi_2}(\vec{u}_1)$, rather than generating an independent $\vec{u}_2$ to compute $V_{\pi_2}(\vec{u}_2)$ (in which case, the covariance term above would be zero).

The use of common random numbers is a classical variance-reduction technique (Reubenstein, 1981; Ross, 1997). In the context of simulation, it is generally recognized that when we are comparing different policies for a randomly-determined system, after the system has been generated, one should then evaluate all the policies for this system. This is essentially the approach recently advanced by researchers in reinforcement learning theory (Ng & Jordan, 2000), though they do not acknowledge the connection between their approach and classical variance-reduction. Their emphasis is on theoretical uniform convergence of value estimates and improved policies derived from a pool of scenarios to the optimal values (their central result is a statement regarding an upper bound, polynomial in the VC-dimension of the policy, on the number of scenarios required to ensure uniform convergence with any stipulated level of probability).

In practical terms, given a pool of $m$ scenarios defined by $\{\vec{u}^{(i)}\}_{i=1}^{m}$, where each $\vec{u}^{(i)}$ provides specific instantiations of random numbers sufficient for simulating a controlled process trajectory, it is a simple matter to compute an estimate of value associated with a parameterized (by $\eta$) policy. For example, we may simply form the sample average of scenario values:

$$V\left(\pi(\eta)\right) \cong \frac{1}{m}\sum_{i=1}^{m} V\left(\pi(\eta), \vec{u}^{(i)}\right).$$

We wish to find the policy parameter, $\eta^*$, that optimizes this quantity. Ng and Jordan consider an example in which the policy-parameter space is a finite set, and in this case it is possible to enumerate exhaustively all possible policies and evaluate them. If the policy-parameter space is compact, as it is for the case of BAMDP's, we may seek an improved policy by means of gradient ascent. Symbolically taking the gradient of both sides of the equation above leads to:

$$\nabla_\eta V\left(\pi(\eta)\right) \cong \frac{1}{m}\sum_{i=1}^{m} \nabla_\eta V\left(\pi(\eta), \vec{u}^{(i)}\right).$$

The term within the sum gauges how the value of a trajectory, implicitly defined by a particular instantiation of random numbers, changes with respect to policy parameters. This gradient is not well-defined for discrete action sets. In simulating stochastic action-selection over a finite set, we form the cumulative distribution function (cdf) from the policy's point mass function over actions, generate a random number uniformly over $[0, 1]$, and map this number through the inverse of the cdf to select the action. The cdf is a "staircase" function with points of discontinuity determined by the parameter $\eta$. For a fixed random number, its inverse image through the cdf (*i.e.* the random number's associated action) may change discontinuously as we vary $\eta$.

Still, we may gain some sense of parametric sensitivity by perhaps adopting finite-difference approximations to the gradient; that is, we could consider repeatedly and systematically perturbing some nominal $\eta$ to $\eta'$ and computing the corresponding scenario-based measure of $V(\pi(\eta'))$. The $\Delta V$ differences between the values of nominal and perturbed policies could serve as a basis for hill-climbing in the space of policy parameters.

The development of more robust Monte-Carlo techniques geared specifically for gradient estimation is another possible direction for future work. In addition to conventional variance reduction techniques, we could consider algorithms that have been developed for sensitivity analysis of Markov chains (Cao, in preparation; Cao & Chen, 1997; Cao & Wan, 1998) and gradient estimation for general stochastic systems (Glynn, 1990; Reiman & Weiss, 1989). Recent research in reinforcement learning theory (Greensmith, Bartlett, & Baxter, 2001) explores the application of a classical variance reduction technique, the use of "control variates," to reduce the variance of gradient estimates produced by policy gradient methods, and is particularly relevant to our concerns.

The tabular stochastic policy iteration scheme presented in the first part of Chapter 4 possesses nice convergence properties (it may be globally-convergent; see the footnote on page 126). The key aspects of the algorithm that make this possible are that the stochastic policy is represented exactly by a table and that the gradient with respect to controller parameters is projected onto the tangent subspace generated by active constraints. In passing to an approach in which we represent policies by parameterized function approximators, we no longer need a projection step to enforce feasibility of our policies, but projecting the gradient in some way may still be a good idea. For example, we may attempt to project the gradient in a way that conceptually mimics the one-step greedy policy-improvement step of conventional policy iteration (see Section 4.1.3). We recall that conventional policy iteration improves the policy via:

$$\pi(s) = \arg\max_a \left\{ r(s,a) + \gamma \sum_{s'} p^a_{ss'} \widetilde{V}(s') \right\} \qquad \forall s \in S.$$

If we adopt a parameterized representation of policies (parameter vector $\eta$), we may attempt to imitate this locally-greedy policy-improvement step by moving in a direction suggested by $\frac{\partial \widetilde{V}(s)}{\partial \eta}$. Since $\eta$ influences the policy over all states, greedily changing $\eta$ to improve $\widetilde{V}(s)$ may decrease $\widetilde{V}(s')$ at other states $s' \neq s$. To mimic conventional policy improvement, we may consider "regularizing" or projecting $\frac{\partial \widetilde{V}(s)}{\partial \eta}$ in such a way that resulting parameter changes have minimal side-effects upon other $\widetilde{V}(s')$'s.

## 7.3 Extended conjugate families, sparse priors, and uncertain reward

We have adopted the matrix beta (Dirichlet) distribution as a convenient natural conjugate distribution in our analysis. This results in a model for uncertainty in which rows of the generalized transition matrix, $P$, are mutually independent random vectors. Martin (1967) has advanced the idea of "extended natural conjugate distributions," which admit nonzero correlation between rows of $P$. These distributions may be constructed by scaling the kernel of the likelihood function by any nonnegative Borel function (*e.g.*, a function that is continuous at all but a finite number of points). There are thus an unlimited number of distinct conjugate families of distributions, and the Borel function gives us considerable latitude in choosing the shapes of the priors. Moreover, as Martin shows (p. 147) the Borel function can be chosen in such a way that it incorporates statistical dependency between rows of $P$.

For situations in which we have very little prior information concerning state transition probabilities, in particular, situations in which we do not even know which transitions are possible, it is possible to define a generalized prior that accounts for possible alternative MDP topologies. For each state, Friedman and Singer's (1999) "sparse multinomial prior" starts from a prior over the size of the set of possible successor states, then assumes that all successor sets of the same cardinality have the same prior probability. Given a successor set, they define a Dirichlet prior over unknown transition probabilities in which all Dirichlet parameters have a common value; i.e., $m_i = m \; \forall i$. Friedeman and Singer then show how to update the priors in light of new observations; the result is an uncertainty model that is the usual Dirichlet model, assuming that only observed transitions are possible, but scaled to account for the possibility of unobserved, novel transitions.

We have concentrated on modeling uncertainty in transition probabilities, but it is possible to apply the Bayesian framework to model uncertain reward distributions as well. If it is known that rewards are drawn from some finite set, then we can proceed as we did previously for uncertain transition probabilities, assuming that rewards have a multinomial distribution with unknown parameters modeled by multivariate beta distributions. If the set of possible rewards has infinite cardinality, then we may adopt conjugate families appropriate to the reward set range. For example, Silver (1963) suggests that if the reward range is $(0, \infty)$, then it may be convenient to adopt an exponential distribution for modeling rewards. The corresponding conjugate family for modeling uncertainty in the exponential distribution's parameter is the family of gamma distributions. Similarly, if the reward range is $(-\infty, \infty)$, then reward may be modeled by a normal distribution, with associated conjugate families of distributions for modeling unknown parameters taken to be the normal or normal/gamma families, depending upon whether the variance is assumed to be known or unknown, respectively.

Let us consider the case of uncertain reward, where possible rewards are drawn from some finite set, $r \in \mathcal{R} = \{r_1, r_2, \ldots, r_K\}$. Our model supposes that, given a transition, $s \xrightarrow{a} s'$, the reward is multinomially-distributed over the possible $r_k$, $k = 1, 2, \ldots, K$. Our uncertainty in the multinomial parameters (probabilities for each $r_k$ associated with the $s \xrightarrow{a} s'$ transition) is characterized by a multi-variate beta (Dirichlet) distribution with parameters, ${}^k n^a_{ss'}$, $k = 1, 2, \ldots, K$. The entire collection of these parameters, for all $k, s, s'$, and $a$, characterize our uncertainty in reward distributions associated with all possible transitions, and may be regarded as the "reward information state" for our problem (this information state component has a total of $K/A//S/^2$ elements). We may now form a new expanded hyperstate, $[s, M, N]$, whose components are physical state, transition-probability information-state, and reward information-state.

Let $n^a_{ss'} = \sum_k {}^k n^a_{ss'}$ and let $m^a_s = \sum_{s'} m^a_{ss'}$. We may write Bellman's equation for optimal value as:

$$V(s, M, N) = \max_a \left\{ \sum_{s'} \sum_k \frac{m^a_{ss'}}{m^a_s} \frac{{}^k n^a_{ss'}}{n^a_{ss'}} \left[ r_k + \gamma V \left( s', M(m^a_{ss'} + +), N({}^k n^a_{ss'} + +) \right) \right] \right\}$$

where, for example, $N\left({}^k n^a_{ss'} + +\right)$ denotes the matrix of reward information-state Dirichlet parameters, $N$, with its ${}^k n^a_{ss'}$th parameter incremented by 1.

To make matters more concrete, consider a problem with two states, two actions, and two possible values of reward, $r \in \{r_1, r_2\}$. Information state components may be written in matrix form:

$$M = \begin{bmatrix} m^1_{11} & m^1_{12} \\ m^1_{21} & m^1_{22} \\ m^2_{11} & m^2_{12} \\ m^2_{21} & m^2_{22} \end{bmatrix} \qquad N = \begin{bmatrix} {}^1n^1_{11} & {}^2n^1_{11} \\ {}^1n^1_{12} & {}^2n^1_{12} \\ {}^1n^1_{21} & {}^2n^1_{21} \\ {}^1n^1_{22} & {}^2n^1_{22} \\ {}^1n^2_{11} & {}^2n^2_{11} \\ {}^1n^2_{12} & {}^2n^2_{12} \\ {}^1n^2_{21} & {}^2n^2_{21} \\ {}^1n^2_{22} & {}^2n^2_{22} \end{bmatrix},$$

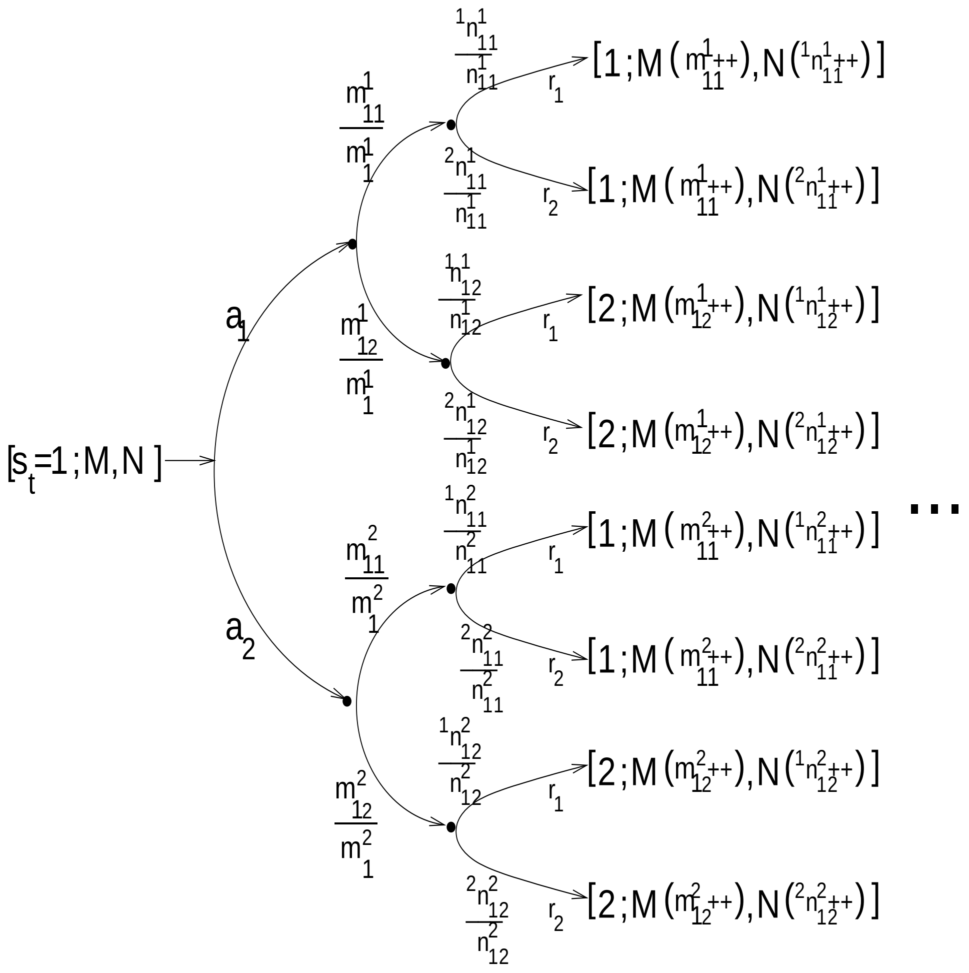


Figure 7.1: The hyperstate transition diagram for a BAMDP with uncertain reward distributions.

and the hyperstate transition diagram is depicted in Figure 7.1. We may proceed to consider the computation of approximately-optimal policies for this extended BAMDP using, for example, Chapter 4's policy-gradient algorithm. In this case, our feature vector becomes our previously-defined feature vector of transition-probability information state elements augmented by the feature vector of reward-information state elements.

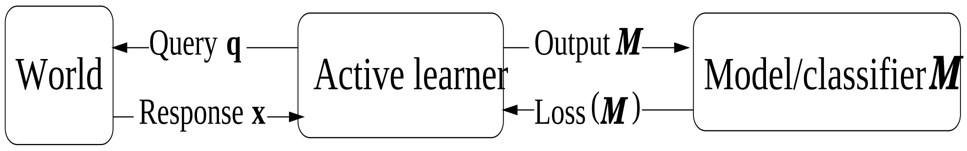


Figure 7.2: An active learner adaptively chooses its queries to minimize model loss.

## 7.4 Hyperopic active learning

In many supervised and unsupervised machine learning tasks, we proceed by gathering a sizable pool of data, randomly sampled from the underlying population distribution, then use the data to train a classifier or model. Typically, the most time-consuming and expensive step in this procedure is the gathering of data, and we thus seek ways to minimize the number of data instances. The process of guiding the sampling process by querying for certain types of instances based upon the data that we have seen so far is called *active learning* (Cohn, Ghahramani, & Jordan, 1996; Tong, 2001; Roy & McCallum, 2001)—see Figure 7.2.

Given that we have chosen some specific model, $\mathcal{M}$, appropriate for our learning task, we define a measure of model quality, or model loss, $Loss(\mathcal{M})$. We then choose as our next query the one that will result in a future model with lowest model loss.

If we were to submit a specific query, $q$, eliciting a response $x$, then this new data would result in an updated, posterior model, $\mathcal{M}'$, and an associated loss. We can gauge the potential of a query, without actually submitting it, by computing the *expected* model loss after asking the query, where the expectation is over all possible query responses. That is, we gauge the effect of submitting query $q$ by

$$Loss(q) = E_x Loss(\mathcal{M}').$$

At each step of active learning, we consider each potential query and compute the expected posterior model loss. Active learning algorithms are typically myopic—they choose to pose the query with lowest expected posterior model loss.

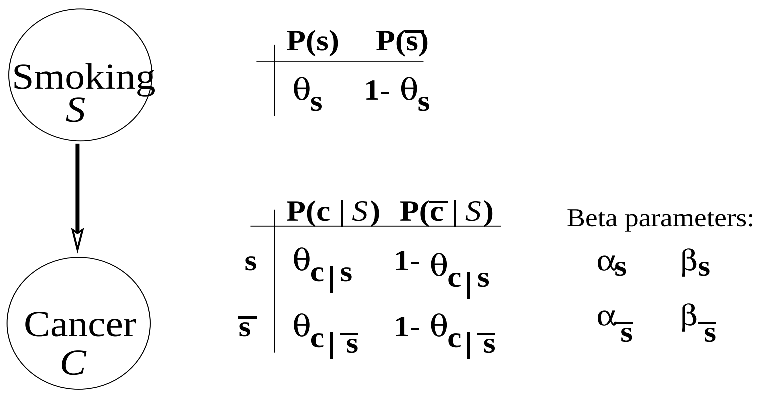


Figure 7.3: A simple Bayesian network ($s$ and $\bar{s}$ respectively denote "smoker" and "non-smoker"; $c$ and $\bar{c}$ denote "cancer" and "not-cancer").

Active learning approaches have been applied to problems of classification, regression, and function optimization. We shall briefly examine its application to the problem of parameter estimation in a very simple Bayesian network, and we will consider how our optimal-learning techniques can lead to a tractable, far-sighted approach.

Consider the simple Bayesian network depicted in Figure 7.3, which represents the structural dependence between two random variables, $S$ (smoker/non-smoker) and $C$ (cancer/no-cancer). At a finer level of detail, dependence is characterized by tables of parameters for conditional distributions (which are Bernoulli in this case). Our Bayesian framework models uncertainty in these parameters by corresponding beta distributions.

We assume that we are presented with some pool of smokers and non-smokers, and our goal is to refine our estimates for conditional distribution parameters by adaptively choosing a subject for thorough medical examination. Active learning takes parameter uncertainty distributions (or equivalently, these distributions' parameters, $M$) and selects a query, $q \in \{\text{smoker}, \neg\text{smoker}\}$, then uses the resulting complete instance $(q, x)$ to update the conditional distribution parameter uncertainties.

Our measure of model quality is based upon the "risk" of Bayesian point estimates. If the true value of conditional probability parameters is $\theta^*$ and our estimate is $\widetilde{\theta}$, then we incur some loss, $Loss(\widetilde{\theta}, \theta^*)$. We may choose the parameter loss function in different ways, squared-error loss and log-loss are viable choices, but we shall adopt the relative entropy, or Kullback-Leibler divergence as our measure:

$$KL(\widetilde{\theta}, \theta^*) = \sum_x P_{\theta^*}(x) \log_2 \frac{P_{\theta^*}(x)}{P_{\widetilde{\theta}}(x)}.$$

We do not have access to the true $\theta^*$, however we do have our uncertainty distributions (beta distributions), which we write as $p(\theta^*)$, that characterize our beliefs about the likely value of $\theta^*$, and we may define the *risk* of a particular $\widetilde{\theta}$ with respect to $p(\theta^*)$ as:

$$E_{\theta^* \sim p(\theta^*)} KL(\widetilde{\theta}, \theta^*) = \int_{\theta^*} KL(\widetilde{\theta}, \theta^*) p(\theta^*) d\theta^*.$$

For all the loss functions we have mentioned, the Bayesian point estimate (the value of $\widetilde{\theta}$ that minimizes the risk relative to $p$) is simply the mean value of the parameters; *i.e.*, $\widetilde{\theta} = E_{\theta^* \sim p(\theta^*)} \theta^*$. If we then insert this value of $\widetilde{\theta}$ into the expression for risk, then risk becomes a function only of the uncertainty distribution $p(\theta^*)$. For our choice of parameter loss function, the risk becomes the expected KL-divergence of $\widetilde{\theta}$ from $\theta^*$.

We adopt the risk associated with our uncertainty distribution $p(\theta^*)$ as our measure of model quality. A greedy approach seeks to obtain an instance, $(q, x)$, that minimizes the risk of the resulting posterior $p\left(\theta^* | (q, x)\right)$. We do not know which response will be elicited by a specific query, but we do know that it will be sampled from a distribution induced by our query. Hence, we take our measure of model quality for a prospective query to be the expected posterior risk:

$$EPRisk\left(p(\theta^*) | Q \equiv q\right) = E_{\theta^* \sim p(\theta^*)} E_{x \sim P_{\theta^*}(x | Q \equiv q)} Risk(p(\theta^* | Q \equiv q, x)).$$

A greedy active learner evaluates the expected posterior risk for all queries and selects the one with lowest value.

It shall prove convenient to consider the reduction in risk obtained by asking various queries, rather than evaluating the expected posterior risk directly. Let us define this reduction as:

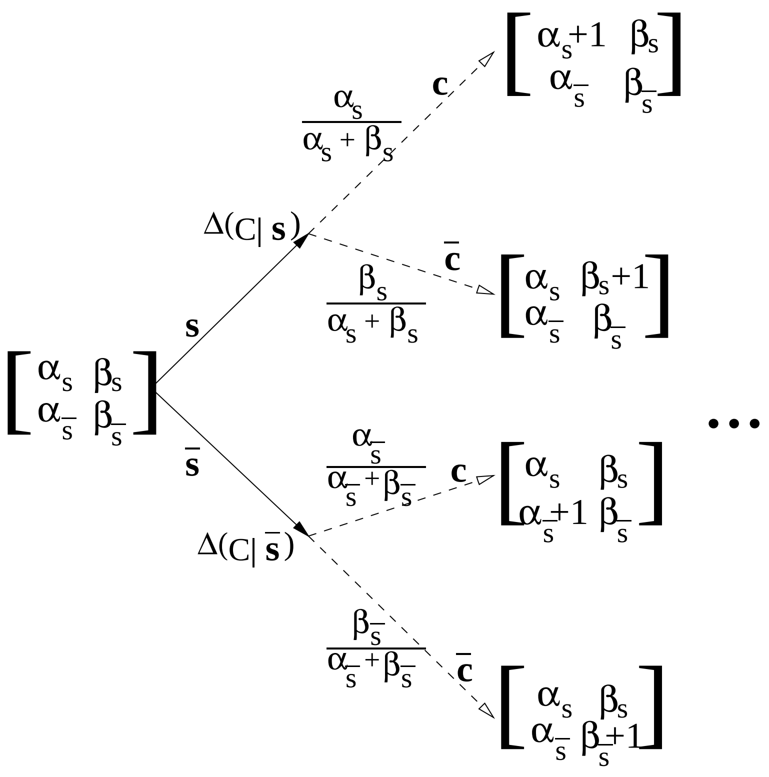
$$\Delta(X|q) = Risk\left(p(\theta^*)\right) - EPRisk\left(p(\theta^*)|q\right).$$

For a simple Bayesian network like our cancer Bayes-net of Figure 7.3, it can be shown (Tong, 2001) that adopting KL-divergence for parameter loss leads to an expression for $\Delta(X|q)$ that is a weighted expected decrement in entropy induced by posing the query. For our cancer Bayes-net this becomes:

$$\begin{aligned}
\Delta(C|q \equiv s) &= p_{\theta^*}(s)\Big\{H\left(\tfrac{\alpha_s}{\alpha_s+\beta_s}, \tfrac{\beta_s}{\alpha_s+\beta_s}\right) \\
&\quad - \Big[\tfrac{\alpha_s}{\alpha_s+\beta_s}H\left(\tfrac{\alpha_s+1}{\alpha_s+\beta_s+1}, \tfrac{\beta_s}{\alpha_s+\beta_s+1}\right) + \tfrac{\beta_s}{\alpha_s+\beta_s}H\left(\tfrac{\alpha_s}{\alpha_s+\beta_s+1}, \tfrac{\beta_s+1}{\alpha_s+\beta_s+1}\right)\Big]\Big\} \\
\Delta(C|q \equiv \bar{s}) &= p_{\theta^*}(\bar{s})\Big\{H\left(\tfrac{\alpha_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}, \tfrac{\beta_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}\right) \\
&\quad - \Big[\tfrac{\alpha_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}H\left(\tfrac{\alpha_{\bar{s}}+1}{\alpha_{\bar{s}}+\beta_{\bar{s}}+1}, \tfrac{\beta_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}+1}\right) + \tfrac{\beta_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}H\left(\tfrac{\alpha_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}+1}, \tfrac{\beta_{\bar{s}}+1}{\alpha_{\bar{s}}+\beta_{\bar{s}}+1}\right)\Big]\Big\}.
\end{aligned}$$

We observe that the uncertainty distribution for variable $S$ is not affected by the selection of a specific query; in the parlance of Bayesian networks, $S$ is not "updatable." We may adopt maximum-likelihood estimates for $p_{\theta^*}(s)$ and $p_{\theta^*}(\bar{s})$, and these estimates will not change with repeated queries. Again, at each stage, a greedy active learner selects the query that maximizes $\Delta$.

In a scenario in which we are allowed to pose a fixed number of sequential queries, we may adopt a more far-sighted approach. The active learning problem may be viewed as a problem of optimal learning; the associated hyperstate transition diagram for the simple cancer/smoking Bayes-net is depicted in Figure 7.4. We note that this problem is quite similar to the optimal probing example presented in Chapter 4, but differs from that case in that (1) Here, there is essentially only one physical state, and we sample possible "transitions" by posing queries, which for this case play the



$\begin{bmatrix} \alpha_s & \beta_s \\ \alpha_{\bar{s}} & \beta_{\bar{s}} \end{bmatrix}$
s
$\bar{s}$
$\Delta(C|\,s)$
$\Delta(C|\,\bar{s})$
$\frac{\alpha_s}{\alpha_s+\beta_s}$
c
$\begin{bmatrix} \alpha_s+1 & \beta_s \\ \alpha_{\bar{s}} & \beta_{\bar{s}} \end{bmatrix}$
$\frac{\beta_s}{\alpha_s+\beta_s}$
$\bar{c}$
$\begin{bmatrix} \alpha_s & \beta_s+1 \\ \alpha_{\bar{s}} & \beta_{\bar{s}} \end{bmatrix}$
$\frac{\alpha_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}$
c
$\begin{bmatrix} \alpha_s & \beta_s \\ \alpha_{\bar{s}}+1 & \beta_{\bar{s}} \end{bmatrix}$
$\frac{\beta_{\bar{s}}}{\alpha_{\bar{s}}+\beta_{\bar{s}}}$
$\bar{c}$
$\begin{bmatrix} \alpha_s & \beta_s \\ \alpha_{\bar{s}} & \beta_{\bar{s}}+1 \end{bmatrix}$
•••

Figure 7.4: Hyperstate transition diagram for our active learning problem.

role of “actions,” and (2) The reward here is related to KL-divergence rather than distribution variance, and reward is weighted by $p(S)$. (We note that this problem is not a bandit problem, since hyperstates are defined in terms of *all* condition probability distribution uncertainty parameters.)

We may consider applying our policy-gradient optimal learning algorithm to this example, just as we did for the optimal probing example in Chapter 4. Our feature vector may be taken to be the set of four beta distribution parameters, and there is no physical state. For this simple example, a far-sighted approach turns out to be unwarranted—the optimal querying strategy chooses to reduce $\Delta$ greedily. This follows from the fact that parameters associated with the $S$ node are not affected by queries, there is no physical state, and incremental reductions of entropy are monotonically decreasing with repeated sampling (just as for the the optimal probing example, where incremental reductions of variance for a fixed physical state are

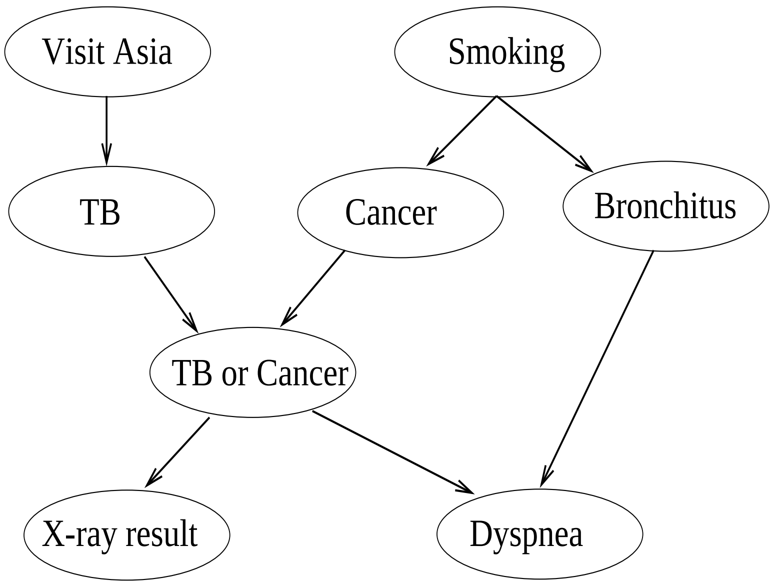


Figure 7.5: A more complex Bayesian network (from Lauritzen & Spiegelhalter, 1988; parameter values available from http://www.norsys.com/netlib/Asia.net). Each of the "control nodes," taken to be VISIT ASIA and SMOKING, can be "clamped" to one of two values, or can be randomly sampled from the true marginal—there are thus nine possible "actions" for each stage of active learning in which the goal is to reduce network KL-divergence.

monotonically decreasing with repeated sampling (see Figure 4.13), though for optimal probing, sampling causes change in physical state). In more richly-structured Bayesian networks in which it is possible to query alternative nodes (Figure 7.5), information may propagate in more complex ways—greedy active learners may not necessarily be optimal, and a far-sighted optimal learning approach would be justified.

## 7.5 The Holy Grail

Many algorithms have been proposed for computing optimal policies for POMDP's, and there may well be approaches that we have not considered that could be extended and applied to BAMDP's. In Chapter 5, we saw how the linear system of constraints defining facets of the value function for POMDP's generalizes to a system of linear-functional inequalities for the BAMDP case. One could consider discretizing the linear-functional system and applying POMDP solution techniques. We have previously observed that Hansen's (1998) interpretation of dynamic programming's policy improvement update as a transformation of a finite-state (deterministic) controller might be extended to controllers represented by stochastic automata (this is an open problem).

The idea of an *analytic* solution for BAMDP's may seem absurd, given the correspondence between POMDP's and BAMDP's presented in Chapter 5 and the complexity of the problem of computing optimal policies for POMDP's,[1] but information-state in the BAMDP case evolves in a relatively "smooth" way that is not mirrored in the POMDP case, and an analytic assault on the optimal learning problem for BAMDP's could make use of this property—our diffusion-modeling approach, summarized in Chapter 6, is one such attempt.

The problem of efficiently computing optimal strategies for bandit problems was an open issue for nearly half a century before Gittins solved it. No known index result exists for problems like ours that possess a physical-state component to their hyperstate, but researchers have attempted to generalize Gittins's method to larger classes of problems (Bertsimas & Nino-Mora, 1996). Perhaps, ultimately, a simple and elegant approach for determining general optimal learning policies will be found.

[1]POMDP's are PSPACE-complete for the finite-horizon case, and are EXPTIME-hard for the infinite-horizon discounted case (the infinite-horizon case may even be undecidable). In somewhat more practical terms, for infinite-horizon POMDP's, the problem of finding the best policy whose size is polynomial in the size of the input is PSPACE-complete (Littman, 1996; Papadimitriou & Tsitsiklis, 1987).

# BIBLIOGRAPHY

[1] Arnold, L. (1974). *Stochastic Differential Equations.* Wiley, New York.

[2] Arrow, K. J., Blackwell, D., & Giershick, M. A. (1949). Bayes and minimax solutions of sequential decision problems. *Econometrica*, *17*, 214-244.

[3] Berry, D. A. & Fristedt, B. (1985). *Bandit Problems: Sequential Allocation of Experiments.* Chapman & Hall, London.

[4] Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, (pp. 30-37). Morgan Kaufmann, San Francisco.

[5] Bartlett, P. L. & Baxter, J. (2000). Estimation and approximation bounds for gradient-based reinforcement learning. *Thirteenth Annual Conference on Computational Learning Theory.*

[6] Bar-Shalom, Y. (1981). Stochastic dynamic programming: Caution and probing. *IEEE Trans. Automatic Control, 26*, 1184-1194.

[7] Bar-Shalom, Y. & Tse, E. (1976). Caution, probing and the value of information in the control of uncertain systems. *Annals of Economic and Social Measurement*, *5*, 323-337.

[8] Barto, A., Sutton, R., & Watkins, C. (1990). Learning and Sequential Decision Making. In M. Gabriel & J. Moore (Eds.), *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, (pp. 539-602). MIT Press, Cambridge, MA.

[9] Barto, A., Bradtke, S., & Singh, S. (1991). *Real-Time Learning and Control Using Asynchronous Dynamic Programming* (Tech. Rep. No. 91-57), University of Massachusetts Amherst, Department of Computer Science.

[10] Barto, A. & Duff, M. (1994). Monte-Carlo Matrix Inversion and Reinforcement Learning. In J. D. Cohen, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems, 6*, (pp. 687-694). Morgan Kaufmann, San Francisco.

[11] Barto, A., Bradtke, S., & Singh, S. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, *72*, 1-138.

[12] Baxter, J. & Bartlett, P. L. (2000). Reinforcement learning in POMDP's via direct gradient ascent. *Proceedings of the Seventeenth International Conference on Machine Learning.*

[13] Bellman, R. (1956). A problem in the sequential design of experiments. *Sankhya*, 16, 221-229.

[14] Bellman, R. (1961). *Adaptive Control Process: A Guided Tour.* Princeton Univ. Press, Princeton, NJ.

[15] Bellman, R. & Kalaba, R. (1959). On adaptive control processes. *IRE Trans.*, *4,* 1-9.

[16] Benveniste, A., Metivier, M., & Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations,* Springer-Verlag, New York.

[17] Bernard, E. (1992). Temporal-difference methods and Markov models. *IEEE Trans. Syst. Man & Cybern.*, *22*, 357-365.

[18] Bertsekas, D. (1987). *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ.

[19] Bertsekas, D. & Castanon, D. (1989). Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Trans. on Automatic Control, 34*, 589-598.

[20] Bertsekas, D. & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA.

[21] Bertsemas, D. & Nino-Mora, J. (1996). Conservation laws, extended polymatroids, and multi-armed bandit problems; a unified approach to indexable systems. *Math Oper. Res.*, *21*, 257-306.

[22] Billingsley, P. (1961). Statistical methods in Markov chains. *Ann. Math. Stat.*, *32*, 12-40.

[23] Blackwell, D. (1965). Discounted dynamic programming. *Ann. Math. Stat. 36*, 226-235.

[24] Bokar, V. & Varaiya, P. (1979). Adaptive control of Markov chains I: finite parameter set. *IEEE Trans. on Automatic Control, 24*, 953-958.

[25] Cao, X. R. (in preparation) From perturbation analysis to Markov decision processes and reinforcement learning.

[26] Cao, X. R. & Chen, H. F. (1997). Potentials, perturbation realization, and sensitivity analysis of Markov processes. *IEEE Trans. on Automatic Control*, *42*, 1382-1393.

[27] Cao, X. R. & Wan, Y. W. (1998). Algorithms for sensitivity analysis of Markov systems through potentials and perturbation realization. *IEEE Trans. Control Systems Technology*, *6*, 482-494.

[28] Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. *Journal of AI Research*, 4, 129-145.

[29] Cover, T. M. & Thomas, J. A. (1991) *Elements of Information Theory.* John Wiley & Sons, New York.

[30] Cozzolino, J. M., Gonzales-Zubieta, R., & Miller, R. L. (1965). *Markovian decision processes with uncertain transition probabilities* (Technical Report No. 11*)*. Research in the Control of Complex Systems. Operations Research Center, MIT.

[31] Cybenko, G., Gray, R., & Moizumi, K. (1997). Q-learning: A tutorial and extensions. *Mathematics of Neural Networks*, Kluwer Academic, Boston.

[32] Dawson, R & Good, I. J. (1957). Exact Markov probabilities from oriented linear graphs. *Ann. Math. Stat.*, *28*, 946-956.

[33] Darken, C. & Moody, J. (1992). Towards faster stochastic gradient search. In J. Moody, S. Hansen, & R. Lippmann (Eds.), *Advances in Neural Information Processing Systems, 4*, (pp. 1009-1016). Morgan Kaufmann, San Francisco.

[34] Dayan, P. & Sejnowski, T. (1996). Exploration Bonuses and Dual Control. *Machine Learning*, *25*, 5-22.

[35] Dean, T. L. & Wellman, M. P. (1991). *Planning and Control.* Morgan Kaufmann, San Mateo, CA.

[36] Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (AAAI-98).

[37] De Bruijn, N. G. & Van Aardenne-Ehrenfest, T. (1950). Circuits and trees in oriented linear graphs. *Simon Stevin*, *4*, 203-217.

[38] DeGroot, M. (1970) *Optimal Statistical Decisions*, McGraw-Hill, New York.

[39] D'Epenoux, F. (1963). Sur un probleme de production et de stochage dans l'aleatoire. *Rev. Fr. Autom. Inf. Recherche Op. 14*; (Engl. trans.) *Man. Sci 10*, 98-108.

[40] Derman, C. & Strauch, R. (1966). A note on memoryless rules for controlling sequential decision processes. *Ann. Math. Stat. 37*, 276-278.

[41] Drake, A. W. (1962). *Observation of a Markov process through a noisy channel*, Ph.D. Thesis*,* Dept. of Electrical Engineering, M.I.T.

[42] Duff, M. O. (1995). Q-learning for bandit problems. In A. Prieditis & S. Russell (Eds.), *Machine Learning: Proceedings of the Twelfth International Conference on Machine Learning* (pp. 209-217). Morgan Kaufmann, San Francisco.

[43] Duff, M. O. & Barto, A. G. (1997). Local bandit approximation for optimal learning problems. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in Neural Information Processing Systems—9* (pp. 1019-1025). MIT Press, Cambridge, MA.

[44] Duff, M. O. (2001). Monte-Carlo algorithms for the improvement of finite-state stochastic controllers: Application to Bayes-adaptive Markov decision processes. *Proc. of AI & Statistics*, Key West.

[45] Fel'dbaum, A. (1965). *Optimal Control Systems*, Academic Press, New York.

[46] Gibbons, J. D. (1985). *Nonparametric Statistical Inference.* Marcel Dekker.

[47] Gittins, J. C. & Jones, D. M. (1974). A dynamic allocation index for the sequential design of experiments. In J. Gani et al., (Eds.), *Progress in Statistics* (pp. 241-266). North-Holland.

[48] Gittins, J. C. & Jones, D. (1979). Bandit processes and dynamic allocation indices (with discussion)," *J. R. Statist. Soc. B 41*, 148-177.

[49] Gittins, J. C. (1989). *Multi-armed bandit allocation indices.* John Wiley and Sons, Chichester.

[50] Glazebrook, K. D. (1980). On Stochastic Scheduling with Precedence Relations and Switching Costs. *J. Appl. Prob. 17*, 1016-1024.

[51] Glazebrook, K.D. & Gittins, J.C. (1981). On single-machine scheduling with precedence relations and linear or discounted costs. *Operations Research, 29*, 289-300.

[52] Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*, 75-84.

[53] Greensmith, E., Bartlett, P. L., & Baxter, J. (2002). Variance reduction techniques for gradient estimates in reinforcement learning. In T. Dietterich, S. Becker, & Z. Ghanramani (Eds.), *Advances in Neural Information Processing Systems—14*, MIT Press, Cambridge, MA.

[54] Gullapalli, V. & Barto, A. (1994). Convergence of indirect adaptive asynchronous value iteration algorithms. *Neural Information Processing Systems, 6*, 695-702.

[55] Hansen, E. A. (1998). *Finite-memory control of partially observable systems.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Massachusetts at Amherst.

[56] Howard, R.A. (1960). *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA.

[57] Howard, R. A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, *2*, 22-26.

[58] Ishikida, T. & Varaiya, P. (1994). Multi-armed bandit problems revisited. *Journal of Optimization Theory & Applic*ations, *83*, 113-154.

[59] Jaakkola, T., Jordan, M. & Singh, S. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation, 6*, 1185-1201.

[60] Jacobs, O. L. R. & Patchell, J. W. (1972). Caution and probing in stochastic control, *Int. J. Control, 16*, 189-199.

[61] Kaelbling, L. P. (1993). *Learning in Embedded Systems.* MIT Press, Cambridge, MA.

[62] Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237-285.

[63] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4, 373-395.

[64] Katehakis, M.H. & Veinott, A.F. (1987) "The Multi-armed Bandit Problem: Decomposition and Computation." *Math. OR.* 12: 262-268.

[65] Kearns, M. & Singh, S. (1998) Near-optimal reinforcement learning in polynomial time. In J. Shavlik, (Ed.), *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 260-268), Morgan Kaufmann, San Mateo, CA.

[66] Kloeden, P.E. and Platen, E. (1992). *Numerical solution of stochastic differential equations*, Springer, Berlin.

[67] Konda, V. & Tsitsiklis, J. (2000). Actor-critic algorithms. In S.A. Solla, T. K. Leen, & K. R. Muller (Eds.), *Advances in Neural Information Processing Systems—12.* MIT Press, Cambridge, MA.

[68] Kumar, P. R. (1983). Optimal adaptive control of linar-quadratic-Gaussian systems. *SIAM J. on Control and Optimization*, *21*, 163-178.

[69] Kumar, P.R. (1985). A survey of some results in stochastic adaptive control. *SIAM J. Control and Optimization*, *23*, 329-380.

[70] Lai, T. L. (1987). Adpative treatment allocation and the multi-armed bandit problem. *Annals of Statistics*, *15*, 1091-1114.

[71] Larsen, R. J. & Marx, M. L. (1986). *An Introduction to Mathematical Statistics and Its Applications.* Prentice-Hall, Englewood Cliffs, NJ.

[72] Lauritzen, S. L. & Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, *50*, 157-194.

[73] Luenberger, D.G. (1973). *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA.

[74] McNamara, J. & Houston, A. (1985). Optimal Foraging and Learning. *Journal of Theoretical Biology*, *117*, 231-249.

[75] MacQueen, J. (1966). A modified dynamic programming method for Markov decision problems. *J. Math. Anal. Appl.*, 14, 38-43.

[76] Marbach, P. & Tsitsiklis, J. N. (2001) Simulation-based optimization of Markov reward processes. *IEEE Trans. Automatic Control, 46,* 191-209.

[77] Martin, J. J. (1967). *Bayesian Decision Problems and Markov Chains*, John Wiley & Sons, New York.

[78] Meuleau, N. & Bourgine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35, 117-154.

[79] Meuleau, N., Kim, K. E., Kaelbling, L. P, & Cassandra, A. R. (1999). Solving POMDPs by searching the space of finite policies. *Proceedings of the Fifteenth International Conference on Uncertainty in AI.*

[80] Monahan, G. E. (1982). A survey of partially observable Markov decision processes: theory, models, and algorithms. *Management Science*, *28*, 1-16.

[81] Moore, A. W. & Atkeson, C. G. (1993). Prioritised sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *35*, 117-154.

[82] Ng, A. & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence.*

[83] Papadimitriou, C. H. & Tsitsiklis, J. N. (1987) The complexity of Markov decision processes. *Mathematics of Operations Research*, 12, 441-450.

[84] Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York.

[85] Peek, N. (1999). Explicit temporal models for decision-theoretic planning of clinical management. *Artifical Intelligence in Medicine*, *15*, 135-154.

[86] Platzman, L. K. (1977). *Finite memory estimation and control of finite probabilistic systems.* Ph. D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT.

[87] Puterman, M. L, & Brumelle, S. L. (1978). The analytic theory of policy iteration. In M. L. Puterman (Ed.), *Dynamic Programming and its Application* (pp. 91-114), Academic Press, New York.

[88] Puterman, M. L., & Shin, M. C. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Man. Sci.* , *24*, 1127-1137.

[89] Raiffa, H. & Schlaifer, R. (1961). *Applied Statistical Decision Theor*y. Graduate School of Business Administration, Harvard University, Boston.

[90] Reiman, M. I., & Weiss, A. (1989). Sensitivity analysis for simulations via likelihood ratios. *Operations Research, 37,* 830-844.

[91] Robbins, H. (1952). Some Aspects of the Sequential Design of Experiments. *Bull. Amer. Math. Soc.*, *58*, 527-535.

[92] Robinson, D. R. (1982). Algorithms for evaluating the dynamic allocation index. *Op. Res. Letters, 1*, 72-74.

[93] Rosen, J. B. (1960). The gradient projection method for nonlinear programming: part I, linear constraints. *SIAM J. Applied Mathematics, 8*, 181-217.

[94] Ross, S. (1983). *Introduction to Stochastic Dynamic Programming*, Academic Press, New York.

[95] Roy, N. & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the Intl. Conf. on Machine Learning.*

[96] Satia, J. K. (1968). *Markovian Decision Processes with Uncertain Transition Matrices or/and Probabilistic Observation of States*, Ph. D. Dissertation, Stanford University.

[97] Satia, J. K. & Lave, R. E. (1973). Markov decision processes with uncertain transition probabilities. *Operation Research*, *21*, 728-740.

[98] Sato, M. & Takeda, H. (1988). Learning control of finite Markov chains. *IEEE Transactions on Systems, Man, and Cybernetics*, *18*, 677-684.

[99] Savage, L. J. (1954). *The Foundations of Statistics*, John Wiley & Sons, New York.

[100] Silver, E. A. (1963). *Markovian decision processes wih uncertain transition probabilities or rewards* (Technical Report No. 1). Research in the Control of Complex Systems, Operations Research Center, MIT.

[101] Singh, S. P. (1994). *Learning to Solve Markovian Decision Processe*s. Ph.D. thesis, University of Massachusetts, Amherst.

[102] Singh, S. P. & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22*, 123-158.

[103] Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvari, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning, 38*, 287-308.

[104] Smallwood, R. D. & Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*, 1071-1088.

[105] Sutton, R. S. (1988) Learning to Predict by the Method of Temporal Differences. *Machine Learning 3*, 9-44.

[106] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 216-224). Morgan Kaufmann, San Mateo, CA.

[107] Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

[108] Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In S.A. Solla, T. K. Leen, & K. R. Muller (Eds.), *Advances in Neural Information Processing Systems—12* (pp. 1057-1063). MIT Press, Cambridge, MA.

[109] Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning, 8*, 257-277.

[110] Thompson, W.R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika, 25*, 275-294.

[111] Thrun, S. B. (1992). The role of exploration in learning control. In D. A. White & D. A. Sofge (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Van Nostrand Reinhold.

[112] Tong, S. (2001). *Active Learning: Theory and Applications.* Ph. D. thesis, Dept. of Computer Science, Stanford University.

[113] Tse, E. & Bar-Shalom, Y. (1973). Wide-sense adaptive dual control of stochastic nonlinear systems. *IEEE Trans. Automatic Control*, *18*, 109-117.

[114] Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning, 16*, 185-202.

[115] Tsitsiklis, J. N. & Van Roy, B. (1997). Approximate solutions to optimal stopping problems. In M. Mozer, M. I. Jordan, & T. Petsche, (Eds.), *Advances in Neural Information Processing Systems—9* (pp. 1082-1088). MIT Press, Cambridge, MA.

[116] Varaiya, P., Walrand, J. & Buyukkoc, C. (1985). Extensions of the multi-armed bandit problem: The discounted case. *IEEE Trans. Automatic Control, 30*, 426-439.

[117] Wald, A. (1947). S*equential Analysis.* John Wiley & Sons, New York.

[118] Walrand, J. (1988). *An Introduction to Queueing Networks.* Prentice Hall, Englewood Cliffs, NJ.

[119] Watkins, C. J. H. (1989). *Learning from Delayed Rewards.* Ph.D. thesis, King's College, Cambridge, UK.

[120] Weber, R. (1992). On the Gittens Index for Multi-armed Bandits. *Annals of Applied Probability*, 1024-1033.

[121] Weiss, G. (1988). Branching bandit processes. *Probab. Engng. Inform. Sci., 2*, 269-278.

[122] Werbos, P. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook, 22*, 25-38.

[123] White, D. J. (1985). Real applications of Markov decision processes. *Interfaces, 15*, 73-83.

[124] White, D. J. (1988). Futher real applications of Markov decsion processes. *Interfaces 18,* 55-61.

[125] Whiteheard, S. D. (1991). Complexity and cooperation in Q-learning. In *Proceedings of the Eighth International Workshop on Machine Larning*, pp. 363-367.

[126] Whittle, P. (1955). Some distribution and moment formulae for the Markov chain. *J. Roy. Stat. Soc., Ser. B., 17*, 235-242.

[127] Whittle, P. (1980). Multi-armed bandits and the Gittins index. *J. Royal Statist. Soc. Ser. B, 42*, 143-149.

[128] Whittle, P. (1981). Arm-acquireing bandits. *Ann. Prob.*, 9, 284-292.

[129] Whittle, P. (1982). *Optimization over Time: Dynamic programming and Stochastic Control, Vol. 1.* Wiley, New York.

[130] Wiering, M. & Schmidhuber, J. (1998). Efficient model-based exploration. In R. Pfeiffer, B. Blumber, J. Meyer, and S. Wilson, (Eds.), *Proceedings of the Sixth International Conference on the Simulation of Adaptive Behavior: From Animals to Animats, 6.*

[131] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229-256.

[132] Wonham, W. M. (1969). Optimal stochastic control. *Automatica*, 5, 113-118.

[133] Wyatt, J. L. (2001). Optimistic model selection for exploration control. To appear in *Proceedings of the 18th International Conference on Machine Learning.*