Decision making with inference and learning methods

by

Matthew William Hoffman

B.S., University of Washington, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF SCIENCE
(Computer Science)

The University Of British Columbia
(Vancouver)

March 2013

Abstract

In this work we consider probabilistic approaches to sequential decision making. The ultimate goal is to provide methods by which decision making problems can be attacked by approaches and algorithms originally built for probabilistic inference. This in turn allows us to directly apply a wide variety of popular, practical algorithms to these tasks. In Chapter 1 we provide an overview of the general problem of sequential decision making and a broad description of various solution methods. Much of the remaining work of this thesis then proceeds by relying upon probabilistic reinterpretations of the decision making process. This strategy of reducing learning problems to simpler inference tasks has been shown to be very fruitful in much of machine learning, and we expect similar improvements to arise in the control and reinforcement learning fields.

The approaches of Chapters 2–3 build upon the framework of [Toussaint and Storkey, 2006] in reformulating the solution of Markov decision processes instead as maximum-likelihood estimation in an equivalent probabilistic model. In Chapter 2 we utilize this framework to construct an Expectation Maximization algorithm for continuous, linear-Gaussian models with mixture-of-Gaussian rewards. This approach extends popular linear-quadratic reward models to a much more general setting. We also show how to extend this probabilistic framework to continuous time processes. Chapter 3 further builds upon these methods to introduce a Bayesian approach to policy search using Markov chain Monte Carlo. In Chapter 4 we depart from the setting of direct policy search and instead consider value function estimation. In particular we utilize least- squares temporal difference learn-

ing to reduce the problem of value function estimation to a more standard regression problem. In this chapter we specifically tackle the use of regularization methods in order to encourage sparse solutions. In Chapters 5–6 we consider the task of optimization as a sequential decision problem. In the first of these chapters we introduce the bandit framework and discuss a number of variations. Then in Chapter 6 we discuss a related approach to optimization utilizing Bayesian estimates of the underlying, unknown function. We finally introduce a novel approach to choose between different underlying point selection heuristics.

Preface

Much of the work of this thesis, unless otherwise noted, was done under the supervision of Nando de Freitas and Arnaud Doucet.

The work of Chapter 3 was originally based off a preliminary idea of Ajay Jasra for applying Markov Chain Monte Carlo to the problem of policy search. This idea became the technical report [Hoffman et al., 2007b] and with more improvements was transformed into [Hoffman et al., 2007a]. The later extensions of this chapter formed the basis for improving upon the earlier MCMC methods in [Hoffman et al., 2009c] which was joint work with Hendrik Kück.

We then noticed that we could use these same ideas to attack the problem of linear-Gaussian control with much more general reward functions, the ideas for which were used in [Hoffman et al., 2009b] and are what Chapter 2 is based on. The later sections of this chapter are based off work with Nando de Freitas on extending the earlier maximum-likelihood framework to semi-Markov control problems in what became [Hoffman and de Freitas, 2011]. Peter Carbonetto also provided a great deal of help on an earlier presentation of this work [Hoffman et al., 2009a].

In the winter of 2011, I took a leave from UBC to join the SEQUEL group at INRIA, Lille where I worked with Mohammad Ghavamzadeh, Alessandro Lazaric, and Rémi Munos on problems of sparse approximations to Least-Squares Temporal Difference methods. There I contributed to work in [Ghavamzadeh et al., 2011], which does not form part of this thesis. Ultimately the results of this collaboration became [Hoffman et al., 2011b] which forms the basis of Chapter 4.

The work of Chapter 6 was joint work with Eric Brochu on combining Bayesian optimization with methods built upon portfolios of experts. The code for this section was written by Eric, and the portfolio strategy itself came about through many discussions. I also provided the additional convergence theory. Finally, this earlier work with Eric (and the influence of the INRIA group) led to further study of bandit approaches. Based on this, the unpublished work of Chapter 5 resulted from additional investigation of bandit methods with Bobak Shahriari.

Table of Contents

List of Figures

Acknowledgments

I'd first and foremost like to thank my supervisors, Nando de Freitas and Arnaud Doucet, for all their time and advice. Nando for his extremely varied interests and push to try new things; Arnaud for his almost encyclopedic knowledge of all things Monte Carlo. And the less said about Arnaud's going-away parties the better.

A great many thanks go to all my collaborators, whose names certainly pepper this work and without whom this wouldn't have been possible. This goes double for all of the students at UBC with whom I've spent many hours at cruddy local pubs (and a few not so bad), running to catch that last bus in the rain, or late-night runs to the Village for deadline food. This time wouldn't have been the same without you. I'll take the cowards' way out and not name names for fear of leaving anyone out, but you all know who you are!

Also, a whole heaping of thanks go to my parents for all their help over the years that I've been working towards this. Both for that and for only occasionally asking me when, exactly, I'd be graduating. The same goes for my brother, Patrick and his new little addition, Myla—she hasn't done much yet, but surely learning my name is help enough.

And finally, saving the best for last, I'd like to thank Tülin for all her help and support over the years. You've certainly done more than your share in helping me work towards finishing this thing, and more so than anyone else I couldn't have done it without you.

Chapter 1

Introduction

The problem of decision making under uncertainty is pervasive, with many different applications. Consider, for example, the task of online path planning. In this setting, a robot must make movements in the world subject to uncertainty both in its own position as well as the position of various landmarks around it. Deciding where to move next then depends not only on the goals of the robot, but also in how the robot can move in order to gain information about its surroundings. Ultimately, by gaining better information about its surroundings the robot may be able to obtain more precise knowledge that will help it better attain its goals in later steps. See e.g. the work of [Martinez-Cantin et al., 2007, 2009]. Another great example of the decision making process is that of low-level motor control. Industrial applications often involve robots which must perform precise motion tasks, however various movements entail differing levels of uncertainty that the robotic system must account for; see [Peters and Schaal, 2007, 2008]. Often these tasks can be learned in an offline manner, but the holy-grail of robotics involves developing robots with the ability to act autonomously and react appropriately to changes in their environment. Another much cited and celebrated example of such control is that of learning a controller for autonomous helicopter flight; see [Ng et al., 2006].

Even problems in computer vision are beginning to rely more heavily on planning approaches. For example, consider the problem of active vision,

i.e. a system which actively decides where to "look" in order to properly maintain track of an object. If the system is limited—either physically or computationally—in such a way that it cannot attend to every point in its visual field, then it must decide where in this field to look [e.g. Bazzani et al., 2011, Denil et al., 2012]. See also the work of [Ballard and Hayhoe, 2009, Rothkopf and Ballard, 2010] for work on task-based visual saliency, particularly for the task of online navigation. One can also consider the "reverse" of planning, i.e. observing another agent and attempting to deduce their goals. Although this approach need not be directly related to vision, there is a great deal of literature in cognitive science on the problems of action understanding and goal inference, see [Ullman et al., 2009]. See also [Ng and Russell, 2000].

The field of operations research also devotes a great deal of effort to the design of computational strategies—often based on dynamic programming—for solving ubiquitous decision making problems. See [Puterman, 1994] for an extensive introduction to this area; see also [Bellman, 1957]. Applications abound in finance, including option pricing in computational finance [e.g. Tsitsiklis and Van Roy, 2001]. More recently, applications in environmental studies and energy management have received a lot of attention; [e.g. Crowley and Poole, 2011, Hannah and Dunson, 2011]. Medical applications are another large consumer of dynamic programming techniques; see [Sauré, 2012]. In fact the scheduling of medical tests may be one of the first big applications, albeit simple, of decision making processes of this type [Robbins, 1952, Thompson, 1933]. In this setting the goal is to assign treatments to a sequence of patients in order to maximize the well-being of all incoming patients. However, the efficacy of each treatment is *a priori* unknown, and as a result it is often beneficial to *explore* the space of possible treatments rather than purely *exploiting* the current best, but possibly sub-optimal treatment.

As we will also claim later, many problems of exploration for sequential optimization can also be seen as decision problems. In fact the online assignment of treatments to medical patients can be seen as related to an optimization process, where the goal is to find the best possible treatment. However, unlike more pure optimization problems, the decision maker is

more often interested not just in finding the best treatment, but is instead interested maximizing the well-being of *all* patients. We will see later in this work that it is possible to clarify the distinction between these two settings. Recently similar approaches have been used to optimize the parameters of other machine learning algorithms [Snoek et al., 2012]; see also the earlier work of [Brochu et al., 2007, 2010a] which applied these techniques to the problem of automating graphics and animation design.

Ultimately, the goal of this work is to propose and develop probabilistic approaches to the problem of planning under uncertainty. This approach, which as we will see has already had considerable success for planning in Markov Decision processes, broadens the scope of solution methods capable for these problems by allowing the full range of machine learning methods to be brought to bear. We will begin by introducing Markov decision processes (MDPs) in the next section we will show in Chapter 2 how to transform this problem into an alternative probabilistic model and perform maximum likelihood estimation for the parameters of this model. The result of this transformation, based on earlier work of [Toussaint and Storkey, 2006], is that it provides an easy framework to apply existing machine learning and statistical methods. For example, the mixture-of-Gaussians model that we introduce in Chapter 2 will be made much simpler to derive based on its relation to standard hidden Markov models.

While the expectation-maximization (EM) algorithm we derive in Chapter 2 is related to other, more standard, dynamic programming algorithms, the probabilistic approach we advocate also allows for much more diverse solution approaches. For example, in Chapter 3 we will introduce a Bayesian solution method based on Markov chain Monte Carlo (MCMC). This is a significant departure from standard methods based on dynamic programming and has immediate applications as a way of performing gradient-free optimization of the associated MDP. However, it also provides interesting avenues for future research as a way of combining model-based and model-free methodology, which we will briefly allude to in the conclusion of that chapter. We will also later show a more sample-efficient Bayesian method for optimizing similar problems in a gradient-free manner in Chapter 6, an

approach which relies on placing a probabilistic model over the space of parameters being optimized.

Finally, we need not look too far to see other examples of the benefits obtained by blurring the lines between standard planning approaches and probabilistic models. The approaches we will discuss in Chapter 4 are based on a least-squares interpretation of the problem of value function approximation (which we will introduce in the next section). This insight, originally noted by [Bradtke and Barto, 1996], opens the floodgate for the multitude of methods in machine learning based on minimizing sums of squared-errors. In particular, in this Chapter we will discuss various regularization schemes in order to induce sparse estimates, and thereby accurately learn the value function when many, many features are used in its approximation.

1.1 Markov decision processes

Throughout this work we will consider the standard framework of reinforcement learning and planning [Sutton and Barto, 1998] wherein an agent interacts with a stochastic environment by making decisions at discrete times n. We will model this interaction, unless otherwise noted, as a discrete-time Markov Decision Process (MDP) [Puterman, 1994] consisting of

- an $\mathcal{X}$-valued state process $\{X_n\}_{n\geq 1}$;
- an $\mathcal{A}$-valued action process $\{A_n\}_{n\geq 1}$;
- and a real-valued reward process $\{R_n\}_{n\geq 1}$.

In particular, each round of interaction proceeds as follows:

For each round $n = 1, 2, \ldots$

1. the decision maker arrives in state $x_n \in \mathcal{X}$;
2. the decision maker chooses an action $a_n \in \mathcal{A}$;
3. the decision maker receives reward $r_n \in \mathbb{R}$ from the environment;
4. the environment transitions to state x_{n+1}.

We will assume that the stochastic processes in question are modeled by

- an initial-state distribution with density $\mu(x_1)$;
- a stationary transition distribution with density $f(x_{n+1}|x_n, a_n)$, conditional on the current state and action;
- and a stationary, stochastic policy with conditional density $\pi_\theta(a_n|x_n)$ parameterized by some real-valued vector $\theta \in \Theta \subseteq \mathbb{R}^d$.

Finally, we will also assume deterministic rewards given by

- a reward function $r(x_n, a_n)$.

We should note here that it is trivial to generalize this model to stochastic rewards with density $g(r_n|x_n, a_n)$, however we will generally assume deterministic rewards for simplicity. A graphical model depicting this process can be seen in Figure 1.1. Note that the policy model introduced here does not preclude deterministic policies, which can be encoded as delta masses[1]. Finally, we will assume that μ, f, and π_θ are densities with respect to a suitable dominating measure, generally Lebesgue or counting.

In order to ease notation later we will also note that for a fixed policy π this model induces a Markov process $\{Z_n\}_{n\geq 1}$ over the extended state-space $\mathcal{Z} = \mathcal{X} \times \mathcal{A}$. We will refer to realizations of this chain as *state-action paths*, and write their associated densities as

$$\mu_\theta(z_1) = \mu(x_1)\, \pi_\theta(a_1|x_1), \tag{1.1}$$

$$f_\theta(z_{n+1}|z_n) = f(x_{n+1}|x_n, a_n)\, \pi_\theta(a_{n+1}|x_{n+1}). \tag{1.2}$$

In a slight abuse of notation we will also let $\pi_\theta(z_n) = \pi_\theta(a_n|x_n)$ and $r(z_n) = r(x_n, a_n)$ denote the policy and rewards in terms of "state-action variables" wherever this usage is made clear from the use of "z" terms. Letting $z_{1:k} = (z_1, \ldots, z_k)$ denote a k-length state-action trajectory we can also write the joint density of this path as

$$p_\theta(z_{1:k}) = \mu_\theta(z_1) \prod_{n=2}^{k} f_\theta(z_n|z_{n-1}). \tag{1.3}$$

[1] Either as Dirac or Kronecker delta functions for continuous or discrete state spaces respectively.

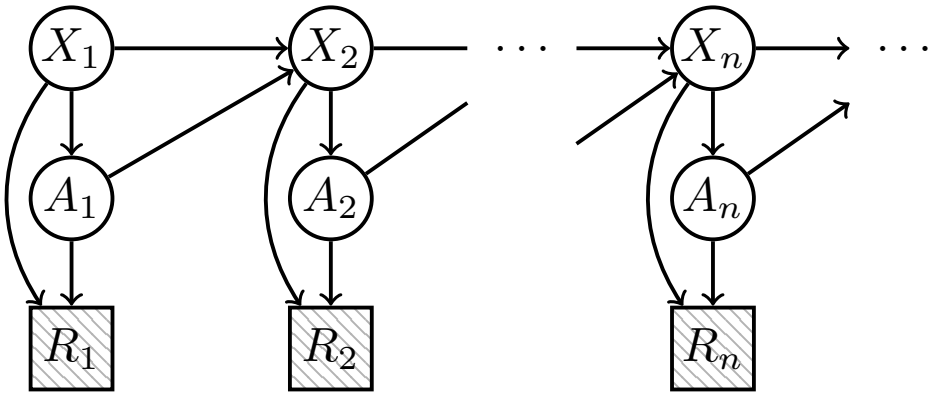

Figure 1.1: Graphical depiction of an MDP, where (X_n, A_n) denote the random variables associated with states and actions at time n and $R_n = r(X_n, A_n)$ is the rewards. Here the shaded square nodes denote the fact that rewards R_n are deterministic given their parents.

Given this formulation we can now turn to the question of what it means for a policy to be "optimal" with respect to this MDP. For some finite k we can consider the finite-horizon expected reward

$$J_k(\theta) = \mathbb{E}_\theta\left[\sum_{n=1}^{k} r(Z_n)\right]$$

where this expectation is taken with respect to the finite dimensional distribution $p_\theta(z_{1:k})$ defined previously. Alternatively, we can consider the infinite-horizon average reward, namely

$$J_\infty(\theta) = \lim_{k\to\infty} \frac{1}{k} J_k(\theta).$$

Finally, given some discount factor $\gamma \in (0, 1)$ we can also introduce the infinite-horizon discounted reward

$$J_\gamma(\theta) = \mathbb{E}_\theta\left[\sum_{n=1}^{\infty} \gamma^{n-1}\, r(Z_n)\right]. \tag{1.4}$$

It is this objective, unless otherwise noted, that we will use for most of Chapters 2–4. As a result we will often write this simply as $J(\theta)$ and leave γ implied. We will also continue to use this objective in the beginning of Chapter 5 and then introduce a slightly modified objective known as regret that is similar to the average reward formulation. We will then use this regret formulation for the final two chapters and will postpone its introduction until those chapters.

Note that the choice of objective is somewhat arbitrary and will depend on the exact goals of the decision maker. The main choice that must be made is between averaged and discounted rewards. By using a discount factor γ we are essentially devaluing later rewards: a reward in n steps will be less valuable by a factor of γ^n. Instead, under the average reward formulation rewards at all time steps are equally valued. Another justification of discounted rewards is that eventually the decision maker will have to stop making decisions at some finite but unknown time. In particular we can consider that there is some probability $1 - \gamma$ that the decision process ends at each step, and otherwise with probability γ it continues. Again, however, we emphasize that this choice lies in the *specification* of the decision problem, whereas in this work we will focus on solving the decision problem once it has been specified. Further, many of the ideas we will introduce can be modified to work in either the discounted or averaged reward settings.

1.2 Solving the decision problem

We can now turn to the problem of how to optimize the objective functions given in the previous section. The decision problems, and the solutions thereof, that we will focus on in this work can be classified on two major axes. The first of these axes is that of *model-based* versus *model-free* scenarios. This distinction boils down to whether or not the models μ, f, and r are known to the learner. The second of these distinctions is based on the form of parameterization used in learning the policy: whether the policy is *directly parameterized* as alluded to in the previous section, or whether it is indirectly parameterized in terms of a *value function*. In this section we will

provide a brief history of different solution methods that fall along these two axes.

Many classical approaches to the problem of decision making are model-based approaches to computing the value function via dynamic programming. Here the "overlapping subproblem" associated with this dynamic programming approach is that of the value function, i.e. a function which given some state returns the long-term value of that state. Given some general policy π we can write the value function as

$$V^\pi(x) = \mathbb{E}_\pi\left[\sum_{n=1}^{\infty} \gamma^{n-1}\, r(X_n, A_n)\middle| X_1 = x\right] \tag{1.5}$$

$$= \mathbb{E}_\pi\left[r(X_1, A_1) + \gamma V^\pi(X_2)\middle| X_1 = x\right]. \tag{1.6}$$

Here we have shown both the direct definition of this value function as an infinite sum of rewards, and the recursive definition which defines the value function in terms of itself. The dynamic programming algorithms we will introduce first generally use this second definition, also known as Bellman's equation [see Bellman, 1957] to define the value function. The key reasoning behind this definition is that we can use an *estimate* of the value function in order to bootstrap towards a better estimate of the value function. In this vein we will introduce the Bellman operator T^π as

$$T^\pi V(x) = \mathbb{E}_\pi\left[r(X_1, A_1) + \gamma V(X_2)\middle| X_1 = x\right] \tag{1.7}$$

which is defined for any function $V(x)$. We can see then that the value function V^π is a fixed point of this operator, i.e. $V^\pi(x) = T^\pi V^\pi(x)$. This immediately suggests an iterative procedure wherein we start with some initial approximation $V^{(0)}$ and successively apply the Bellman operator until convergence. This process will then converge to the unique fixed-point due to the contraction property of the operator—i.e. that each application reduces the error of the approximation, again see [Bellman, 1957] and also [Puterman, 1994]. The computation of this quantity is often known as *policy evaluation*.

Up to this point we have written the Bellman recursion in its most general form, however we can ground this by considering finite state spaces. In this setting, $V^{(i)}$ is actually a vector with dimension given by the size of the state space and application of the Bellman operator can be written in vector form as

$$T^\pi V = r^\pi + \gamma P^\pi V$$

for any vector V. Here r^π is a vector of expected immediate rewards for each state and P^π is a matrix representing the transition model, both under policy π. Note that this type of model is often called *tabular* due to the representation of the transition model as a matrix or *table*. Policy evaluation would then proceed by repeatedly performing this matrix multiplication until each entry of the value vector has converged.

Now, the value function V^π is useful in order assign to each state a value for that state, however we are often more concerned with computing the value of taking an action from any given state. As a result, it is useful to introduce the function $Q^\pi(x, a)$ which denotes the value of taking action a from state x and selecting all further actions by following policy π. This state-action value function can be written as

$$\begin{aligned} Q^\pi(x,a) &= \mathbb{E}_\pi\left[\sum_{n=1}^{\infty} \gamma^{n-1}\, r(X_n, A_n)\middle| X_1 = x, A_1 = a\right] \\ &= r(x,a) + \gamma \mathbb{E}\big[V^\pi(X_2)\big| X_1 = x, A_1 = a\big]. \end{aligned} \tag{1.8}$$

Here we can see that this computation is quite simple if we already have the value function V^π, as the rewards are a deterministic function of (x, a) and we then need only compute the expectation of V^π as a function of the next state X' conditioned on the given state-action pair. The reason for moving to the state-action value function is then simple: we can use it to define a greedy, deterministic policy with respect to the value function, in a step known as *policy improvement*:

$$\pi'(x) = \arg\max_{a \in \mathcal{A}} Q^\pi(x, a).$$

This procedure can then be iterated, alternating between policy evaluation in order to determine V^π (and hence Q^π) and policy improvement to find π'. This process is known as *policy iteration*. Given enough iterations this procedure will converge towards the optimal value function $V^*(x) = \max_\pi V^\pi(x)$ [Puterman, 1994], and as a result we can obtain the optimal policy that is greedy with respect to this value function.

One potential drawback to policy iteration, is that each step involves policy evaluation in order to compute V^π, the convergence of which may only occur in the limit. Alternatively, we can consider a *generalized policy iteration* algorithm which instead iterates between

1. a *value function update* step, which merely improves upon the current value function approximation;
2. a *policy improvement* step which improves the policy by making it greedy with respect to the current value function.

We can see that policy iteration lies inside this more general framework in that the update step performs policy evaluation until convergence. We can, however, consider approaches that only apply the Bellman operator a fixed number of times [see Puterman and Shin, 1978]. At one end of this spectrum lies the approach of *value iteration* which only applies the Bellman update once. We can summarize the update and improvement steps of value iteration by modifying the Bellman operator as follows

$$T^*V(x) = \max_{a \in \mathcal{A}} \big\{ r(x,a) + \gamma \mathbb{E}[V(X_2) | X_1 = x, A_1 = a] \big\}. \tag{1.9}$$

Here we can see that the quantity inside the maximum corresponds exactly to the earlier introduced Bellman operator so long as the policy in question calls for action a—this corresponds to the *update* step. The maximization over a implicitly performs the improvement step. By performing this update for all $x \in \mathcal{X}$ we obtain a single step of value iteration. By combining these two steps we can then consider iteratively computing the fixed point of T^*, i.e. starting from some arbitrary value function $V^{(0)}$ we can compute

$$V^{(i+1)}(x) = T^* V^{(i)}(x)$$

$$= \max_{a \in \mathcal{A}} \big\{ r(x,a) + \gamma \mathbb{E}[V^{(i)}(X')|x,a] \big\}.$$

This approach is known as *value iteration* and like policy iteration, this procedure is known to converge to the optimal value function; again see [Puterman, 1994]. This procedure, can often, although not always, converge much faster.

We can now note how this relates to the objective function $J(\theta)$ introduced earlier. Consider a value function parameterized by some vector θ. For example in the tabular setting we can consider a vector such that $V_\theta(x) = \theta_x$ is the value of the xth state or $Q_\theta(x,a) = \theta_{xa}$ for a matrix of state-action values. By parameterizing these value functions we are *indirectly* parameterizing the policy due to the greedy maximization of this value. We then can write the objective function as

$$\begin{aligned} \theta^* &= \max_\theta J(\theta) \\ &= \max_\theta \mathbb{E}_\theta \left[\sum_{n=1}^{\infty} \gamma^{n-1} r(Z_n) \right] = \max_\theta \mathbb{E}_\theta \big[V_\theta(X_1) \big]. \end{aligned}$$

If the optimal value function V^* exactly computes the value function of the MDP, then θ^* corresponds to the parameterization of this value function. Alternatively we can consider approximating this value function, in which case we must consider the initial-state distribution or some other distribution under which to trade-off the accuracy of the approximation. The field of approximate dynamic programming is primarily concerned with the question of how to approximate these value functions; see [Bertsekas, 1995, Busoniu et al., 2010] for more extensive coverage of this topic.

The approaches described in the preceding paragraphs are, however, only applicable when the MDP model is known. When these models are not known the decision maker is instead faced with the problem of model-free or *reinforcement learning* wherein one can only sample from the MDP of interest. One of the central concepts of reinforcement learning is that of the temporal difference (TD) error. Consider, for example, a learning algorithm that has a current estimate of the value function V, with which it can make

predictions about future rewards. Next we will consider a decision maker that is attempting to learn this quantity online, and from state x takes action a, arriving in state x'. We can see that the quantity $r(x,a)+\gamma V(x')$ is a Monte Carlo approximation to the Bellman operator introduced above. This quantity can then be used to compare the predicted value function to a one-step approximation to the truth, and update the value function via

$$V(x) \leftarrow V(x) + \alpha[r(x,a) + \gamma V(x') - V(x)]. \tag{1.10}$$

We can look at this as a stochastic approximation method for computing the value function, akin to that of [Robbins and Monro, 1951], applied to the dynamic programming approaches previously introduced. The second term consists of a learning rate α multiplied by the TD error, i.e. the error in approximating the value function made by $V(x)$. If actions a are selected according to some policy π, this value function will converge towards V^{π}, and is known as TD(0) [Sutton, 1988]. However, unlike the model-based dynamic programming methods, the state-based value function is not enough to update the policy, and one must instead learn a state-action value function $Q(x,a)$. We must also ensure that all state-action pairs are visited *often enough*, i.e. that there is some non-zero probability of seeing every state and action. The reasoning behind this last requirement is in order to ensure that the MDP is explored in enough detail so that we can properly compare the values of different actions.

The simplest way to ensure these requirements is to use an ϵ-greedy policy based on the current value function $Q(x,a)$. This means that with probability ϵ at every step a random action is taken, and otherwise a greedy action from Q is selected. We can then consider a strategy which from state x takes action a, chosen in an ϵ-greedy manner from Q, transitions to x' and finally selects an ϵ-greedy action a'. We can then introduce the SARSA rule which updates the value function as

$$Q(x,a) \leftarrow Q(x,a) + \alpha[r(x,a) + \gamma Q(x',a') - Q(x,a)]. \tag{1.11}$$

Note, the name of this approach comes from the order of data that is observed by the algorithm, i.e. "state, action, reward, state, action". We can see that this is an extension of the TD-prediction method to the problem of policy learning. Further, we can note that this method is *on-policy* in that the same policy (ϵ-greedy in Q) is used both for the update rule and for exploration. We can, however, extend this method to be *off-policy*, i.e. one following a different strategy for exploration, by writing

$$Q(x,a) \leftarrow Q(x,a) + \alpha[r(x,a) + \gamma \max_{a'' \in \mathcal{A}} Q(x', a'') - Q(x,a)]. \tag{1.12}$$

This strategy is known as Q-learning. Note, the difference in these two formulations is merely in the use of the max for Q-learning. The reasoning behind this off-versus-on policy distinction is that off-policy methods are able to perform more exploration in a way that may hurt their online performance, but instead allows for more quickly learning the optimal value function. For an extensive discussion of these methods see the work of [Sutton and Barto, 1998]; see also [Szepesvári, 2010].

As with the approaches of approximate dynamic programming, it is also possible to use function approximation in the RL setting. These approaches can sometimes diverge when learned off-policy [Baird, 1995], however see the work of [Maei et al., 2010, Sutton et al., 2009] for recent approaches which attempt to surmount this problem. Of particular interest is the use of linear function approximation in which the value function is approximated as $V_\theta(x) = \phi(x)^T\theta$ for some basis features ϕ, an approach known as least-squares TD (LSTD) introduced by [Bradtke and Barto, 1996]. This approach is particular interesting as it allows us to bring in more common approximation techniques from supervised learning. This is a topic that we will return to in more detail in Chapter 4. The work of [Lagoudakis and Parr, 2002] also extends LSTD to the policy iteration setting. See also the work of [Engel et al., 2003, 2005, Rasmussen and Kuss, 2004] for extensions of LSTD to the kernelized setting.

The main criticism of value function methods, however, is still that they are only indirectly learning the policy. A simple policy does not necessarily

translate into an easy to learn value function. Alternatively we can directly parameterize the policy as in the previous section and attempt to directly optimize these parameters, a broad approach we will refer to as *direct policy search*. A widely used method for optimizing this objective involves performing stochastic approximation, again as in the classic approach of [Robbins and Monro, 1951] and following a Monte Carlo estimate of the gradient $\nabla J(\theta)$. We can further take advantage of the particular structure of the MDP objective in writing this gradient. By moving the gradient inside the expectation defining $J(\theta)$, and approximating the infinite integral with some large horizon K we arrive at

$$\begin{aligned}\nabla J(\theta) &= \int \Big[\sum_{n=1}^{K} \gamma^{n-1}\, r(z_n)\Big] \overbrace{\nabla \log p_\theta(z_{1:K})\, p_\theta(z_{1:K})}^{\text{derivative of log}}\, dz_{1:K} \\ &= \int \Big[\sum_{n=1}^{K} \gamma^{n-1}\, r(z_n)\Big] \Big[\sum_{n=1}^{K} \nabla \log \pi_\theta(z_n)\Big] p_\theta(z_{1:K})\, dz_{1:K} \qquad (1.13)\end{aligned}$$

where the marked term in the first equation is purely due to the derivative of the log, i.e. the fact that $\nabla p(x) = \nabla \log p(x)\, p(x)$. A result of this is that so long as we can sample from $p_\theta(z_{1:K})$, i.e. the generative model of the MDP, then we can approximate this gradient. This allows one to perform gradient ascent in either the model-based or model-free scenarios. We should also note that, as pointed out by [Kappen, 2007], the infinite integral in the above equation can also be thought of as a "path integral", a technique more common in the statistical physics literature.

However, the variance of (1.13) can be quite high, depending on the magnitude of the summed rewards for each sampled trajectory. We can now take advantage of the particular structure of this model to reduce the variance. We can first see that due to the two summations in the above equation, our gradient computation consists of reward and policy-gradient components

$$\nabla J(\theta) = \sum_{n=1}^{K} \sum_{t=1}^{K} \mathbb{E}_\theta\big[r(Z_t)\, \nabla \log \pi_\theta(Z_n)\big].$$

Intuitively, for $t < n$ the rewards at time t do not depend on state-action pairs from a later time n, so we would expect these components to average out to zero in expectation. We can formalize this intuition by way of iterated expectation and write

$$\mathbb{E}_\theta\big[r(Z_t)\,\nabla \log \pi_\theta(Z_n)\big] = \mathbb{E}_\theta\Big[r(Z_t)\,\underbrace{\mathbb{E}_\theta\big[\nabla \log \pi_\theta(Z_n)\big|Z_t\big]}_{0}\Big] = 0.$$

Here the marked component is the expectation of a score and as a result this must necessarily be zero. We can then eliminate all these components where $t < n$ and rewrite the gradient as

$$\nabla J(\theta) = \int \Big[\sum_{n=1}^{K} \nabla \log \pi_\theta(z_n) \sum_{t=n}^{K} \gamma^{t-1} r(z_n)\Big] p_\theta(z_{1:K})\, dz_{1:K}. \tag{1.14}$$

This results in the REINFORCE algorithm of [Williams, 1992], developed separately as the GPOMDP algorithm [Baxter and Bartlett, 2001]. It is possible to further reduce this variance by introducing a constant additive term or *baseline* to the reward function, as again the product of this constant term with the expectation of the gradient will be zero. This constant can then be chosen to minimize the variance; see [Greensmith et al., 2001] for more details. We can also greatly speed up convergence of this stochastic approximation by utilizing second-order gradient information, which in the policy optimization setting becomes particular easy to evaluate; see the work of [Peters and Schaal, 2008] for more details. More recently, similar direct policy approaches have been proposed which rely on associating the expected reward $J(\theta)$ with an equivalent probabilistic model. These probabilistic models are constructed in such a way that the maximum of J corresponds to the maximum likelihood estimate of this auxiliary model. We will return to these methods in Chapters 2–3. In those chapters we will focus on model-based direct policy search, however we will point out ways in which they can be used for model-free scenarios.

Finally, we should also mention here the problem of arm selection in k-armed bandit problems [see Robbins, 1952]. This is a reinforcement learning

problem wherein there is no state-space, instead the rewards are stochastically distributed according to some unknown distribution, conditional on a finite set of actions. These actions are often called arms in an analogy to the arms of a slot machine. Ultimately, the goal in the bandit problem is, as in the rest of this section, that of getting the highest sum of rewards. But in this setting, in order to obtain high rewards, one must explore the set of arms in order to gain information about the underlying distribution. If we assume a prior and likelihood model for the rewards of each action we can actually treat the problem of selecting actions as an MDP where the state of this MDP corresponds to the posterior probability distribution of each actions' rewards. The landmark work of [Gittins, 1979] describes an optimal approach to selecting actions based on this MDP formulation. Unfortunately, this procedure can often prove to be quite expensive when planning far into the future. We will further see that there exist many asymptotically optimal exploration strategies for bandit problems which rely only on the current single-step mean and uncertainty for each arm. These approaches provide an interesting way to study the tradeoff between exploration and exploitation, wherein it is possible to approximate the asymptotically optimal policy by being greedy with respect to some other "value function". We will return to this question in Chapters 5–6. We should also note that these approaches are somewhat related to the techniques of Bayesian reinforcement learning [e.g., Poupart et al., 2006], however the lack of an actual state-space obviates many of the difficulties encountered in Bayesian RL.

1.3 Outline of this work and contributions

In the first part of this work we will consider a probabilistic extension of the direct policy search methods briefly alluded to in the previous section. In Chapter 2 we will first introduce a view of the MDP problem as one of inference in an infinite mixture of finite-horizon MDPs and then introduce an Expectation Maximization algorithm for finding the maximum likelihood parameters in this model. We will then use this procedure to develop a linear-Gaussian controller with mixture-of-Gaussians reward models. This

acts as an extension of the popular linear-Quadratic controller—i.e. the most popular industrial controller—to the setting of general reward models. We will also show how to extend the probabilistic interpretation to that of continuous time, semi-Markov decision processes. This first chapter is based off work previously published in [Hoffman and de Freitas, 2011, Hoffman et al., 2009b].

In Chapter 3 we then consider Bayesian approaches to the problem of policy search by extending the models of Chapter 2. The simplest approach would be to introduce a prior or regularizer term to the maximum-likelihood chapters of this earlier chapter. Instead we present what was at the time the first sample-based approach to this problem which produces samples from the "posterior" of the underlying probabilistic model. We present initial algorithms implementing this approach and present initial results demonstrating their effectiveness. In order to apply this approach to more general models, however, we present a number of extensions which greatly improve the mixing time of the Markov chain Monte Carlo (MCMC) algorithms used within this procedure. This chapter is based on work of [Hoffman et al., 2007a, 2009c]. See also the discussion with Kück et al. [2009] on using similar approaches for active sensing and experimental design.

Chapter 4 returns to the problem of value function approximation in an RL setting, and specifically the LSTD approach mentioned earlier. In particular this chapter discussions various regularization approaches that can be applied to this problem in order to encourage sparsity, i.e. the strict selection of only a subset of features. In this chapter an approach based on a mixture of ℓ_2 and ℓ_1 penalties is introduced. This allows one to attack the problem of value function approximation when the number of samples n is much greater than the number of features k. This chapter is based on work of [Hoffman et al., 2011b]; although not discussed, see also work with Ghavamzadeh et al. [2011] analyzing the finite-sample behavior of a related ℓ_1 penalized approach to LSTD.

In Chapter 5 we provide an overview and literature review of multi-armed bandit problems. This serves as an introduction to some of the concepts we will use later in Chapter 6 and serves as a bridge between the approaches

of policy learning and Bayesian optimization. We also provide a number of empirical comparisons between various bandit procedures both in terms of cumulative and simple regret. We also note a number of mildly novel observations based on the performance of these methods on different problem formulations.

Finally, in Chapter 6 we discuss problems of Bayesian optimization, i.e. sequential function optimization in the black-box setting where a posterior over function values is used to drive exploration. In this chapter we note the similarities between Bayesian optimization and the related bandit approaches of the previous chapter. We further proceed to introduce a novel meta-strategy for choosing between Bayesian acquisition strategies. We then show state-of-the-art performance on a number of test problems and highlight the meta-algorithm's ability to correctly choose between various base strategies where it would be otherwise very difficult for even an expert designer to choose appropriately. This chapter is based on work of [Hoffman et al., 2011a].

Chapter 2

Maximum likelihood approaches to solving Markov Decision Processes

In this chapter, we consider the general case of Gaussian mixture reward functions and derive EM updates for this setting. We show that the resulting algorithm outperforms widely used policy gradient approaches. At the heart of our philosophy is the goal of harnessing the power of analytical calculations in conjunction with approximation methods. This strategy has been shown to be very fruitful in inference tasks, and we expect similar improvements to arise in the control and reinforcement learning fields.

Our approach follows from the formulation of [Toussaint and Storkey, 2006] which casts the stochastic control problem instead as one of parameter estimation in a suitable artificial statistical model. From there the authors solve the parameter estimation problem using the Expectation-Maximization (EM) algorithm. The general idea of using inference to solve decision problems appears to have originated in the early work of [Shachter, 1988] on influence diagrams. More closely related to the approach we take in this chapter is the use of EM in [Dayan and Hinton, 1997], however this earlier work only considers the simpler problem of optimizing for immediate rewards. Similar formulations have since been applied to operational

space control [e.g., Peters and Schaal, 2007] and in the sequential setting it has been studied by [Attias, 2003, Verma and Rao, 2006]. Perhaps the most complete and clear formulation is the one of [Toussaint et al., 2006], which presents impressive results for finite state space models, although the authors only consider a single Gaussian reward function.

In this chapter we focus primarily on algorithmic concerns, however we note that these techniques have since enjoyed substantial success in the field of robotics [Kober and Peters, 2008, Peters and Schaal, 2007, Vijayakumar et al., 2009]. A significant body of empirical evidence in these papers also indicates that these methods can often outperform traditional stochastic planning and control methods, as well as more recent policy gradient schemes. The inference duality first noted by Kalman has also been extended recently to more general transition models under the assumption of a very specific form of reward function [see Kappen, 2007, Todorov, 2008]. This work, however, applies mainly to planning domains and may be seen as a specific case of the EM approach. A more general formulation without these earlier reward assumptions was recently proposed by [Rawlik et al., 2012] which clarifies the relationship between these two approaches.

In Section 2.1 of this chapter we introduce the basic formulation of a sequential decision problem as one of statistical inference and outline an EM algorithm for estimating the policy parameters while analytically marginalizing over the states and actions. In Section 2.3 a novel algorithm is introduced using the EM procedure to solve linear MDPs with arbitrary rewards, approximated with mixtures of Gaussians—in other words this extends methods based on linear controllers (i.e. LQR, LQG) to arbitrary reward models. We then proceed in Section 2.4 to demonstrate how well the new algorithm performs on synthetic (but hard) MDPs. We then introduce a motivating robotics example that shows how to map a nonlinear control problem to a linear control problem and transform the reward into one amenable to this approach. Hence, if we can solve linear MDPs with arbitrary rewards, we can attack a large class of difficult nonlinear control problems. Finally, in Section 2.5 we show how to extend this approach to semi-Markov decision processes with continuous time.

2.1 Infinite mixtures of finite-horizon MDPs

By examining Eq. (1.4), we can note that the use of a discount factor γ is very similar to modeling the path length as a geometric random variable K, distributed according to

$$p(k) = (1-\gamma)\gamma^{k-1}. \tag{2.1}$$

In fact this is one of the original justifications for the use of a discount factor, namely that the "world might end" at any time step with probability $1-\gamma$. For reasons that will become clear later, we will also refer to this quantity as the *time prior*. Using this intuition we can write the joint density of both paths and path lengths as

$$p_\theta(k, z_{1:k}) = (1-\gamma)\gamma^{k-1}\, p_\theta(z_{1:k}|k), \tag{2.2}$$

where we will refer to this quantity as the *path prior* with $p_\theta(z_{1:k}|k) = p_\theta(z_{1:k})$ denoting the *k-length path prior*. We should note now that this joint defines a *trans-dimensional* distribution over the space $\bigcup_{k=1}^{\infty}(\{k\} \times \mathcal{Z}^k)$, i.e. a distribution where the dimensionality of the distribution (K in this case) is also a random variable.

Given that we have shown how to incorporate the discount factor into the probabilistic model of an MDP we will now take this one step further and incorporate the immediate reward function $r(z)$. In particular, following the notation of [Toussaint and Storkey, 2006], we will introduce a "dummy variable" R with the following conditional probability

$$p(R = 1|k, z_k) = r(z_k). \tag{2.3}$$

Note that this is the *likelihood* of our "observation" R and as a result it need not integrate to one with respect to z_k. Due to the interpretation of $r(z)$ as the probability associated with observing R, however, this formulation does require that $r(z)$ be positive in order to ensure that the density is well defined. We will later show how to ease this restriction in some situa-

tions. Now, by combining the probability (2.3) with the path prior (2.2) and marginalizing over $(k, z_{1:k})$ we can write the *marginal likelihood* of observing $R = 1$ as

$$p_\theta(R = 1) = \sum_{k=1}^{\infty} \int p_\theta(k, z_{1:k})\, p(R = 1|k, z_k)\, dz_{1:k}. \tag{2.4}$$

As the following proposition will show, we can directly relate this quantity to the expected reward $J(\theta)$.

Proposition 1 (Toussaint et al. 2006)**.** *The objective function $J(\theta)$ is proportional to the marginal likelihood defined in* (2.4)*. More precisely,*

$$J(\theta) = (1 - \gamma)^{-1}\, p_\theta(R = 1). \tag{2.5}$$

Proof. By rescaling the expected reward (1.4) by $(1 - \gamma)$ and exchanging the order of integration and summation we can write

$$\begin{aligned}
(1 - \gamma)J(\theta) &= (1 - \gamma)\mathbb{E}_\theta\left[\sum_{k=1}^{\infty} \gamma^{k-1}\, r(z_k)\right] \\
&= (1 - \gamma) \int \left[\mu_\theta(z_1) \prod_{n=2}^{\infty} f_\theta(z_n|z_{n-1})\right] \left[\sum_{k=1}^{\infty} \gamma^{k-1}\, r(z_k)\right] dz_{0:\infty}, \\
&= \sum_{k=1}^{\infty} \int (1 - \gamma)\, \gamma^{k-1}\, p_\theta(z_{1:k}|k)\, r(z_k)\, dz_{1:k} \\
&= p_\theta(R = 1).
\end{aligned}$$

□

As a result the optimal policy parameters correspond exactly to the maximum likelihood estimate of θ in this statistical model given observations $R = 1$, i.e. $\theta^* = \arg\max_\theta p_\theta(R = 1)$. It is this problem that we will focus on in the rest of this chapter.

A graphical illustration of the full mixture model is shown in Figure 2.1 and at this point we should comment on the *interpretation* of this probabilistic model. We can think of the reward likelihood $p(R = 1|k, z_k)$ as equivalent to the emission probability in a state-space model (SSM)—e.g.

hidden Markov models (HMMs) or linear dynamical systems (LDSs). In other words, given a path of length k which ends in the extended-state z_k this quantity gives the probability of observing $R = 1$. By this same token, we can think think of path instantiations $(k, z_{1:k})$ as equivalent to the hidden states of a standard SSM. This model, however, differs from the standard formulation in two key regards. The first of these differences is simple, namely that there is only one observation $R = 1$ made at the end of the latent trajectory. One benefit of this formulation is that the reward "likelihood" is only evaluated at time K, i.e. we need only compute observation likelihoods that occur at the last time step. This representation will prove particularly useful later when performing inference as we will only need to evaluate this likelihood at the end of the chain and then propagate this backwards in time.

The second key difference with standard SSMs is that both the latent trajectory as well as the length of this trajectory are hidden. Here the discount factor γ induces a distribution which mixes over finite-horizon MDPs whose stopping time is given by the random variable K. This final difference causes no immediate difficulties in working with the resulting marginal likelihood, however we must keep in mind that the expectation in (2.4) is now integrating with respect to a varying number of random variables.

Finally, we should also note that the actual value of the observation $R = 1$ is completely arbitrary. We can see from the proof of Proposition 1 that the only important detail concerning this observation is that the probability of observing $R = 1$ conditioned on a path terminating in the extended-state z_k must be equal to the rewards $r(z_k)$. As a result in the rest of this work we will leave the actual value of the observation implicit and write instead $p(R|k, z_k)$.

2.2 Maximum likelihood policy search via Expectation Maximization

In the previous section we introduced an alternative probabilistic model such that the maximum likelihood estimate of the policy parameters co-

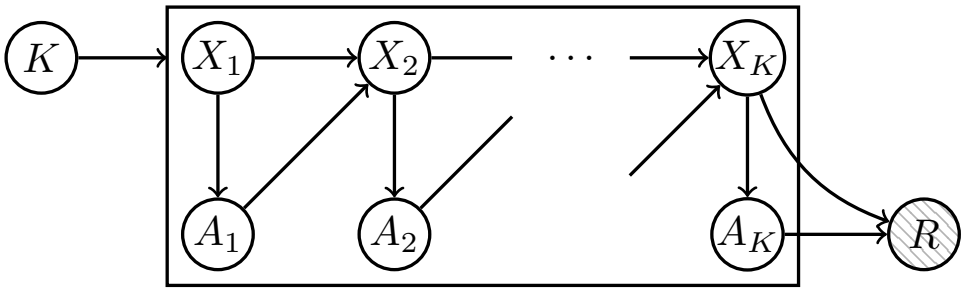

Figure 2.1: Graphical model depicting the problem of policy optimization as inference in an infinite mixture of finite-horizon MDPs. Here the random variable K denotes the random horizon, $Z_n = (X_n, A_n)$ the path components conditioned on this length, and R is a dummy observation whose likelihood is given by the reward $p(R|K, Z_K) = r(Z_K)$.

incides with the optimal policy parameters. In this section we will build on this reformulation by discussing an alternative method of optimizing the MDP objective function. Specifically, we will make use of methods originally developed for maximum likelihood parameter estimation.

Given a particular instantiation of the latent trajectory $(k, z_{1:k})$ we will write the *complete data likelihood* of the parameters θ as

$$\mathcal{L}(\theta|k, z_{1:k}, R) \triangleq p_\theta(k, z_{1:k}, R) = p_\theta(k, z_{1:k})\, p(R|k, z_k). \tag{2.6}$$

This likelihood function is "complete" since it depends on both the observed and unobserved variables. We should emphasize, however, that the "data" in this complete likelihood corresponds only to the dummy observation of $R = 1$. Also note that we will write the likelihood in the form $\mathcal{L}(\theta|\cdots)$ in order to emphasize that this is a function of θ. We do not, however, want to optimize the parameters θ for any one particular trajectory. Instead, as noted earlier, we will treat the random variables $(K, Z_{1:K})$ as hidden and instead optimize the marginal or *incomplete data likelihood* which we repeat

below as

$$\mathcal{L}(\theta|R) \triangleq p_\theta(R) = \sum_{k=1}^{\infty} \int p_\theta(k, z_{1:k})\, p(R|k, z_k)\, dz_{1:k}. \tag{2.7}$$

Note that we have now integrated out all sources of randomness, i.e. the latent variables, and we want to optimize this quantity as a function of θ in order to find the parameters that make our observations most likely.

Directly optimizing this quantity is problematic, however. The joint distribution, $p_\theta(k, z_{1:k})$ does not factorize over the individual states so we cannot treat this as an independent product of the individual z_n. Further, consider the integral in (2.7). For k-length paths containing M discrete state-action pairs this integral would result in a summation over M^k terms, i.e. this summation is exponential in the length of the path. In order to avoid this computationally complex scenario we must instead find another way to maximize the incomplete likelihood. In the rest of this chapter we will do so by making use of an Expectation Maximization procedure which we will introduce shortly.

In order to discuss optimizing the incomplete data likelihood we will first introduce the *path posterior*, i.e. the posterior distribution of our latent variables given our observations

$$p_\theta(k, z_{1:k}|R) = \frac{p_\theta(k, z_{1:k})\, p(R|k, z_k)}{p_\theta(R)}. \tag{2.8}$$

For a fixed θ this quantity represents the distribution over the latent trajectory $(K, Z_{1:K})$ conditioned on the dummy observation R. We can also note that the numerator of this distribution is given by the complete data likelihood and its normalizing constant given by the incomplete data likelihood. We can now introduce an EM algorithm for optimizing the incomplete data likelihood. The notation necessary to the describe this procedure can be summarized as

complete data likelihood: $\mathcal{L}(\theta|k, z_{1:k}, R) = p_\theta(k, z_{1:k}, R)$,
incomplete data likelihood: $\mathcal{L}(\theta|R) = p_\theta(R)$,

posterior distribution: $p_\theta(k, z_{1:k}|R)$.

The algorithm is an iterative procedure that starts from some initial point θ' in parameter space and in the *E-step* constructs the posterior distribution over the latents given this setting of parameters. This distribution can then be used to compute the expected log of the complete data likelihood as a function of θ, i.e.

$$Q(\theta, \theta') = \mathbb{E}_{\theta'}\big[\mathcal{L}(\theta|K, Z_{1:K}, R)\big|R\big].$$

The *M-step* then involves maximizing $Q(\theta, \theta')$ with respect to θ. This process is then repeated until convergence, setting θ' to the parameter estimate obtained at the last iteration. It is well known that this iterative technique is guaranteed to produce a local maximum of the incomplete data likelihood [see e.g. Dempster et al., 1977, McLachlan and Krishnan, 1997].

In order to compute this quantity we will first expand the complete data likelihood from (2.6) as

$$\begin{aligned} p_\theta(k, z_{1:k}, R) = p(k)\,\mu(x_1)\,\pi_\theta(a_1|x_1) \\ \left[\prod_{n=2}^{k} f(x_n|x_{n-1}, a_{n-1})\,\pi_\theta(a_n|x_n)\right] p(R|k, z_k) \end{aligned}$$

from which we can see that if we collect and treat as constant all terms not depending on θ that this is proportional to $\prod_{n=1}^{k}\pi_\theta(z_n)$. Now, taking the logarithm of this quantity and plugging it into the $Q(\theta, \theta')$ we can write

$$\begin{aligned} Q(\theta, \theta') &= \sum_{k=1}^{\infty}\int p_{\theta'}(k, z_{1:k}|R)\ \log p_\theta(k, z_{1:k}, R)\,dz_{1:k} \\ &= \sum_{k=1}^{\infty}\int p_{\theta'}(k, z_{1:k}|R)\left[\sum_{n=1}^{k}\log\pi_\theta(z_n)\right] + \text{const.} \\ &= \sum_{k=1}^{\infty} p_{\theta'}(k|R)\sum_{n=1}^{k}\int p_{\theta'}(z_n|k, R)\ \log\pi_\theta(z_n)\,dz_n + \text{const.} \end{aligned} \tag{2.9}$$

where we have ignored additive constants not depending on θ. In the follow-

ing two sub-sections will describe the resulting E-step needed to construct these posterior distributions and the M-step which will maximize this quantity with respect to θ.

2.2.1 The E-step

In the E-step we will construct the distributions necessary to evaluate the Q-function, ignoring additive constants. By referring to (2.9) we can see that the distributions $p_\theta(k|R)$ and $p_\theta(z_n|k, R)$ must be computed using the parameter estimates from a previous iteration. Given a fixed k we now seek to efficiently compute the marginal over z_n, for which we will use an approach similar to the well-known *forward-backward* algorithm for Hidden Markov Models (HMMs) and Linear Dynamical Systems (LDSs). We will then show that we can easily obtain the posterior distribution of times k given these quantities.

Our first step will be to introduce *forward messages*, which in traditional expositions of the forward-backward algorithm are used to represent the marginal distribution over z_n given all observations that occurred before time n. In our case, observations only occur at the end of a k-length trajectory so we can write this quantity simply as $p_\theta(z_n|k)$ for any $n \leq k$. Note however, that this quantity can be obtained from the path prior $p_\theta(z_{1:k}|k)$ by first integrating out $z_{n+1:k}$. As a result, we can easily see that this density does not directly depend on k, or rather that $p(z_{1:n}|k)$ has the same density for any k so long as $n \leq k$. As a result we can drop the dependence on the horizon k and write

$$\begin{aligned}\alpha_n(z_n) = p_\theta(z_n) &= \int \mu_\theta(z_1) \prod_{j=2}^{n} f_\theta(z_j|z_{j-1})\, dz_{1:n-1} \\ &= \int \alpha_{n-1}(z_{n-1})\, f_\theta(z_n|z_{n-1})\, dz_{n-1}. \end{aligned} \tag{2.10}$$

We can also see from the first line of this formulation that messages are initialized with the initial-state distribution, $\alpha_1(z_1) = \mu_\theta(z_1)$.

Next, we will introduce *backward messages*, which again we can note are

traditionally used to denote the probability of all observations from time $n+1$ onwards given the latent variable z_n. Unlike in the HMM setting, however, we will take advantage of the fact that our model has only one observation at the last time step. Instead we will introduce messages $\beta_\tau(z_n)$ which denote the probability of making observation R in τ time steps starting from the state-action pair z_n. This quantity can be written recursively as

$$\begin{aligned}\beta_\tau(z_n) &= p_\theta(R|k = n+\tau, z_n) \\ &= \int r(z_{n+\tau}) \prod_{j=n}^{n+\tau-1} f_\theta(z_{j+1}|z_j)\, dz_{n+1:n+\tau} \\ &= \int \beta_{\tau-1}(z_{n+1})\, f_\theta(z_{n+1}|z_n)\, dz_{n+1}. \end{aligned} \tag{2.11}$$

We can also easily see from the first line that these messages are initialized with the immediate reward, $\beta_0(z_k) = p(R|k, z_k) = r(z_k)$. This decomposition is useful because it decouples the the backward messages from the path length k. Rather than having to compute this quantity for the tuple of values (n, k), we need only compute this message as a function of a single quantity, the time-to-go τ. As a result, unlike in an HMM-context the forward and backward messages can be computed independently of each other. This was first observed by [Toussaint et al., 2006], and allows us to compute these distributions in parallel using a single pass for each message.

The messages introduced above now provide us with an efficient means of calculating the distributions required for the E-step. We can see that the product of forward and backward messages gives us the following density

$$\begin{aligned} p_\theta(z_n|R, k) = \frac{p_\theta(z_n, R|k)}{p_\theta(R|k)} &\propto p_\theta(z_n|k)\, p_\theta(R|k, z_n) \\ &= \alpha_n(z_n)\, \beta_{k-n}(z_n). \end{aligned} \tag{2.12}$$

Note that the normalizing constant $p_\theta(R|k)$ is independent of n and can be obtained by integrating (2.12) with respect to any z_n. We can then compute

the *time posterior* using Bayes rule, i.e.

$$p_\theta(k|R) \propto p_\theta(R|k)\, p(k). \tag{2.13}$$

Corollary 1 (Toussaint et al. 2006)**.** *The k-step reward is given by integrating the product of forward and backward messages for any n and the infinite-horizon reward is given by the expectation of this quantity with respect to k, i.e.*

$$\mathbb{E}_\theta[r(Z_k)|k] = p_\theta(R|k) = \int \alpha_n(z)\, \beta_{k-n}(z)\, dz \tag{2.14}$$

$$\mathbb{E}_\theta[r(Z_K)] = p_\theta(R) = \sum_{k=1}^{\infty} p(k)\, \mathbb{E}_\theta[r(Z_k)|k]. \tag{2.15}$$

A Monte Carlo E-step

At this point we should also note, that it is also possible to perform a Monte Carlo approximation during the E-step in order to optimize these problems when either the necessary distributions are unknown or the updates cannot be computed in closed form. As noted in [Vlassis and Toussaint, 2009], we can sample from the initial-state and transition distributions in order to approximate the Q-function. Given M trajectories $\{z_{1:k}^{(i)}\}_{i \leq M}$ sampled from $p_\theta(z_{1:k})$ we can approximate the n-step distribution for any $n < k$ with

$$\widetilde{p}_{\theta'}(z_n|k) \approx \frac{1}{p_\theta(R|k)} \cdot \frac{1}{M} \sum_{i=1}^{M} r(z_k^{(i)})\, \delta_{z_n^{(i)}}(z_n),$$

where the normalizing constant $p_\theta(R|k)$ can be approximating by summing over the k-step rewards. If we assume some maximum time-horizon $K_{\max}$ we can approximate the Q-function as

$$Q(\theta, \theta') \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{K_{\max}} \sum_{n=1}^{k} p(k)\, r(z_k^{(i)}) \log \pi_\theta(z_n^{(i)})$$

$$= \frac{1}{M} \sum_{i=1}^{M} \sum_{n=1}^{K_{\max}} \log \pi_\theta(z_n^{(i)}) \sum_{k=n}^{K_{\max}} p(k)\, r(z_k^{(i)}).$$

This function can then be optimized using the same techniques as in the standard M-step. We also note that the noise in this approximation can be reduced by using a reward model which utilizes trajectories of rewards rather than just the reward at the end of the k-length chain. We will return to this later in Section 3.4.1, however this was independently noted by [Vlassis and Toussaint, 2009].

2.2.2 The M-step

The M-step requires us to maximize the Q-function with respect to the policy parameters θ. If possible we can analytically maximize this function by solving for the fixed point of $\nabla_\theta Q(\theta, \theta') = 0$. If this is not possible we can still evaluate the gradient at the current set of policy parameters $\nabla_\theta Q(\theta', \theta')$ and follow the resulting ascent direction, resulting in a generalized EM algorithm (GEM). When this procedure is iterated, both of these methods are known to locally maximize the incomplete data likelihood [again, for more details see McLachlan and Krishnan, 1997].

A related approach involves calculating the "step" that the standard EM algorithm would take, i.e.

$$\widetilde{\nabla} Q(\theta, \theta') = (\arg\max_\theta Q(\theta, \theta')) - \theta' \tag{2.16}$$

and using this value as if it were a gradient. Such an approach was suggested by [Jamshidian and Jennrich, 1993] wherein it was used in the context of a conjugate gradient method; see also the work of [Lange, 1995] who uses a quasi-Newton to speed up this procedure. Finally, although the term in Equation (2.16) is referred to as a generalized-gradient in the cited literature, we will instead refer to it as a *pseudo-gradient* so as not to confuse it with generalized EM. We will illustrate both the GEM and the pseudo-gradient methods later in Section 2.4.

It is also possible to make a direct connection between EM-based algo-

rithms and the policy gradient, $\nabla_\theta J(\theta)$. By rearranging terms we can write the Q-function's gradient as

$$\begin{aligned}\nabla_\theta Q(\theta, \theta') &= \sum_{k=1}^{\infty} \int p_{\theta'}(k, z_{1:k}|R)\, \nabla_\theta \log p_\theta(k, z_{1:k}, R)\, dz_{1:k} \\ &= \sum_{k=1}^{\infty} \int \frac{p_{\theta'}(k, z_{1:k})\, r(z_k)}{p_{\theta'}(R)} \cdot \frac{\nabla_\theta\, p_\theta(k, z_{1:k})}{p_\theta(k, z_{1:k})}\, dz_{1:k}\end{aligned}$$

and by evaluating this gradient at θ', we obtain

$$\begin{aligned}\nabla_\theta Q(\theta', \theta') &= \frac{1}{p_{\theta'}(R)} \int \nabla_\theta\, p_{\theta'}(k, z_{1:k})\, r(z_k)\, dz_{1:k} \\ &= \frac{1}{p_{\theta'}(R)} (1 - \gamma)\, \nabla_\theta J(\theta'),\end{aligned}$$

where the second line follows directly from Equation (2.5). This equivalence is not entirely surprising in light of Proposition 1. However, the derivation is surprisingly clean and allows us to map directly between EM approaches and standard policy gradient approaches. Additionally, it does verify for us that a GEM algorithm as introduced above computes a gradient in exactly the same direction as the policy gradient.

Further, while EM methods are, in general, only able to guarantee local convergence it can be shown via its relation to policy iteration that these methods exhibit global convergence for discrete models when using exact/analytic inference (see the work of [Toussaint et al., 2006] for more details). In more general continuous settings no such guarantees can be made, however as we will see in later results, a sufficiently exploratory initial policy does seem to have a tempering effect. This is especially true if the exact EM updates can be used, as additional exploratory noise does not cause the dramatic increase in variance associated with sample-based methods (such as policy gradients).

Ultimately, the methods used to optimize the EM objective boil down to computing the gradient of Q. By combining the forward and backward

messages from the previous section with (2.9) we can write this gradient as

$$\begin{aligned}
\nabla_\theta Q(\theta, \theta') &= \sum_{k=1}^{\infty} p_{\theta'}(k|R) \sum_{n=1}^{k} \int p_{\theta'}(z_n|k, R)\, \nabla_\theta \log \pi_\theta(z_n)\, dz_n \\
&= \sum_{k=1}^{\infty} \frac{p_{\theta'}(R|k)\, p(k)}{p_{\theta'}(R)} \sum_{n=1}^{k} \int \frac{\alpha_n(z_n)\, \beta_{k-n}(z_n)}{p_{\theta'}(R|k)}\, \nabla_\theta \log \pi_\theta(z_n)\, dz_n \\
&= \frac{1}{p_{\theta'}(R)} \sum_{k=1}^{\infty} p(k) \sum_{n=1}^{k} \int \alpha_n(z_n)\, \beta_{k-n}(z_n)\, \nabla_\theta \log \pi_\theta(z_n)\, dz_n
\end{aligned} \tag{2.17}$$

As a result, in order to compute this gradient we need only compute the integral of the log policy with respect to the product of forward and backward messages. This also provides some intuition as to how to extend this approach to negative rewards. Obviously the forward messages will be a proper positive density, but in allowing for negative rewards it is possible that the backward messages correspond to a density term with respect to an underlying negative measures. We can account for this, however, by breaking the $\beta_\tau(z)$ term into positive and negative components. This idea is similar in spirit to the Hahn-Jordan decomposition of signed measures; see the work of [Poyiadjis et al., 2005] for an example of using signed measures for estimating the gradient of a general state-space model. We will also see an example of this in the next section when dealing with mixture-of-Gaussian rewards.

We can also see from (2.17) that in general the computational complexity of computing this gradient is $O(k_{\max}^2)$ per iteration for some maximum time-horizon $k_{\max}$. In the discrete, tabular case this complexity can be reduced to linear time as noted by [Toussaint and Storkey, 2006]; we will provide more detail on this in Section 2.5. More generally, in certain situations this can also be done in linear time for policies whose components are linear functions the underlying state process [see Furmston and Barber, 2011]; we will also return to this point in Section 2.6.

Up to this point we have presented the EM approach to policy search

in full generality. In the next section we will introduce a particular linear-Gaussian model and show how this results in very efficient updates while still allowing for quite general reward models.

2.3 A mixture of Gaussians model

We will now consider state and action spaces given by $\mathcal{X} = \mathbb{R}^{n_x}$ and $\mathcal{A} = \mathbb{R}^{n_a}$ and a linear-Gaussian[1] transition model and policy,

$$\begin{aligned}\mu(x_1) &= \mathcal{N}(x_1; \mu_1, \Sigma_1),\\ f(x_{n+1}|x_n, a_n) &= \mathcal{N}(x_{n+1}; Ax_n + Ba_n, \Sigma),\\ \pi_\theta(a_n|x_n) &= \mathcal{N}(a_n; Kx_n + m, \sigma^2 I).\end{aligned}$$

Here the policy is parameterized by $\theta = (K, m, \sigma)$, the model itself is parameterized by $(\mu_1, \Sigma_1, A, B, \Sigma)$, and I is the identity matrix. The extended state-space $\mathcal{Z} = \mathbb{R}^{n_x+n_a}$ will be given by by stacking state and action vectors $z = [x; a]$, and we can write the transition models in this space as

$$\mu_\theta(z_1) = \mathcal{N}(z_1; \bar{\mu}_1, \bar{\Sigma}_1), \tag{2.18}$$

$$f_\theta(z_{n+1}|z_n) = \mathcal{N}(z_{n+1}; \bar{F}z_n + \bar{m}, \bar{\Sigma}). \tag{2.19}$$

Although the parameters $(\bar{\mu}_1, \bar{\Sigma}_1, \bar{F}, \bar{m}, \bar{\Sigma})$ depend on θ we have left this dependency implicit in order to simplify the notation. The exact form of the parameters is given in Figure 2.2 and we give these terms without proof as they are relatively simple to derive. We will further assume a reward model which is a combination of P unnormalized Gaussians, i.e. squared exponential functions of the form:

$$r(z) = \sum_{j=1}^{P} w_j \exp\big\{(y_j - M_j z)^T L_j^{-1} (y_j - M_j z)\big\} \tag{2.20}$$

each parameterized by (w_j, y_j, M_j, L_j). It should be emphasized that these functions are only used for their functional form, and in particular each y_j is

[1] Let $\mathcal{N}(x; \mu, \Sigma)$ denote a Normal distribution in x with mean μ and covariance Σ.

State-action transition parameters:

$$\bar{F} = \begin{bmatrix} A & B \\ KA & KB \end{bmatrix} \quad \bar{m} = \begin{bmatrix} 0 \\ m \end{bmatrix} \quad \bar{\Sigma} = \begin{bmatrix} \Sigma & \Sigma K^T \\ K\Sigma & K\Sigma K^T + \sigma^2 I \end{bmatrix}$$

Initial state-action parameters:

$$\bar{\mu}_1 = \begin{bmatrix} \mu_1 \\ K\mu_1 + m \end{bmatrix} \quad \bar{\Sigma}_1 = \begin{bmatrix} \Sigma_1 & \Sigma_1 K^T \\ K\Sigma_1 & K\Sigma_1 K^T + \sigma^2 I \end{bmatrix}$$

Figure 2.2: Definition of the transition parameters for a mixture of Gaussians MDP.

a parameter and should not be interpreted as a random variable. It is also worth noting that even were it normalized this is not *strictly* a Gaussian density in z because of the presence of M_j.

With our model specified, we can now write the forward and backward messages for this problem:

$$\alpha_n(z) = \mathcal{N}(z; \hat{\mu}_n, \hat{\Sigma}_n), \tag{2.21}$$

$$\beta_\tau(z) = \sum_j \exp\{-\tfrac{1}{2}(\check{c}_\tau^j + z^T \check{\Omega}_\tau^j z - 2z^T \check{\mu}_\tau^j)\}. \tag{2.22}$$

The full recursive definition can be seen in Figure 2.3. The updates for the forward message parameters are relatively simple, and are essentially the same as those given in the update phase of the discrete-time Kalman filter. The derivation of the backward messages is a more complicated (and tedious) process, see Appendix A. Here we have reparameterized the individual Gaussian components of the reward model in *canonical form*.

For any k and n we will let $\tau = k - n$ be the time to go. Following from (2.12) we can write the unnormalized marginal posterior distribution over z_n as the product of forward and backward messages

$$p_{\theta'}(z_n|k, R) \propto \alpha_n(z_n)\, \beta_\tau(z_n) = \sum_j \tilde{w}_{n\tau}^j\, \mathcal{N}(z_n; \tilde{\mu}_{n\tau}^j, \tilde{\Sigma}_{n\tau}^j), \tag{2.23}$$

Forward message recursion:

$$\hat{\mu}_1 = \bar{\mu}_1 \qquad \hat{\Sigma}_1 = \bar{\Sigma}_1$$
$$\hat{\mu}_n = \bar{F}\hat{\mu}_{n-1} + \bar{m} \qquad \hat{\Sigma}_n = \bar{F}\hat{\Sigma}_{n-1}\bar{F}^T + \bar{\Sigma}$$

Backward message recursion:

$$\begin{aligned}
\tilde{\Sigma}_\tau^{-1} &= \check{\Omega}_\tau^j + \bar{\Sigma}^{-1} \\
\check{\mu}_1^j &= M_j^T L_j^{-1} y_j \\
\check{\mu}_\tau^j &= \bar{F}^T\bar{\Sigma}^{-1}(\tilde{\Sigma}_{\tau-1}\bar{\Sigma}^{-1}\bar{m} + \tilde{\Sigma}_{\tau-1}\check{\mu}_{\tau-1}^j - \bar{m}) \\
\\
\check{\Omega}_1^j &= M_j^T L_j^{-1} M_j \\
\check{\Omega}_\tau^j &= \bar{F}^T(\bar{\Sigma}^{-1} - \bar{\Sigma}^{-1}\tilde{\Sigma}_{\tau-1}\bar{\Sigma}^{-1})\bar{F} \\
\\
\check{c}_1^j &= -2\log w_j + y_j^T L_j^{-1} y_j \\
\check{c}_\tau^j &= \check{c}_{\tau-1}^j + \log|\bar{\Sigma}\tilde{\Sigma}_{\tau-1}^{-1}| + \bar{m}^T\bar{\Sigma}^{-1}\bar{m} \\
&\quad - (\check{\mu}_{\tau-1}^j + \bar{\Sigma}^{-1}\bar{m})^T\tilde{\Sigma}_{\tau-1}(\check{\mu}_{\tau-1}^j + \bar{\Sigma}^{-1}\bar{m})
\end{aligned}$$

Figure 2.3: The forward and backward message recursions for the mixture of Gaussians MDP. In order to ease notation as much as possible we mark the statistics of the forward messages with a *hat* (e.g. $\hat{a}$) and mark the backward pass statistics with a *check* (e.g. $\check{a}$).

$$\begin{aligned}
\tilde{\Sigma}_{n\tau}^j &= (\hat{\Sigma}_n^{-1} + \check{\Omega}_\tau^j)^{-1} \\
\tilde{\mu}_{n\tau}^j &= \tilde{\Sigma}_{n\tau}^j(\hat{\Sigma}_n^{-1}\hat{\mu}_n + \check{\mu}_\tau^j) \\
\tilde{w}_{n\tau}^j &= |\hat{\Sigma}_n^{-1}\tilde{\Sigma}_{n\tau}^j|^{\frac{1}{2}} \exp\Big\{-\tfrac{1}{2}\big[\check{c}_\tau^j + \hat{\mu}_n^T\hat{\Sigma}_n^{-1}\hat{\mu}_n - (\tilde{\mu}_{n\tau}^j)^T(\tilde{\Sigma}_{n\tau}^j)^{-1}(\tilde{\mu}_{n\tau}^j)\big]\Big\}
\end{aligned}$$

Figure 2.4: The marginal posterior distribution parameterization for the mixture of Gaussians MDP.

where the parameters $(\tilde{w}^j_{n\tau}, \tilde{\mu}^j_{n\tau}, \tilde{\Sigma}^j_{n\tau})$ are defined in Figure 2.4. Further, by integrating this quantity over z_n we can obtain the following quantities

$$p_{\theta'}(R|k) = \int \alpha_n(z_n)\, \beta_\tau(z_n)\, dz_n = \sum_j \tilde{w}^j_{n\tau} \triangleq \tilde{w}_{n\tau}, \tag{2.24}$$

$$p_{\theta'}(k|R) \propto (1-\gamma)\gamma^{k-1}\, \tilde{w}_{n\tau}. \tag{2.25}$$

Again, as noted in Section 2.2.1 the k-step likelihood attains the same value when computed for any n and the time posterior follows directly from (2.13).

Given $\theta = (K, m, \sigma)$ we can now calculate the partial derivatives of $\log \pi_\theta(u|x)$ with respect to each policy parameter and plug these directly into Equation (2.17) to obtain $\nabla_\theta Q(\theta, \theta')$, i.e.

$$\frac{\partial Q}{\partial K} = \sum_k p_{\theta'}(k|R) \sum_{n=1}^{k} \sigma^{-2}\, \mathbb{E}_{\theta'}[A_n X_n^T - m X_n^T - K X_n X_n^T | k, R], \tag{2.26}$$

$$\frac{\partial Q}{\partial m} = \sum_k p_{\theta'}(k|R) \sum_{n=1}^{k} \sigma^{-2}\, \mathbb{E}_{\theta'}[A_n - K X_n - m | k, R], \tag{2.27}$$

$$\frac{\partial Q}{\partial \sigma} = \sum_k p_{\theta'}(k|R) \sum_{n=1}^{k} \sigma^{-3}\, \mathbb{E}_{\theta'}[C^T C | k, R] - n_u \sigma^{-1}, \tag{2.28}$$

where $C = A_n - K X_n - m$. By setting this gradient equal to 0 and solving for θ we can obtain the EM update. The expectations needed to perform these calculations can be trivially obtained from the sufficient statistics of Z_n, i.e.

$$\mathbb{E}_{\theta'}[Z_n | k, R] = \tilde{w}^{-1}_{n\tau} \sum_j \tilde{w}^j_{n\tau}\, \tilde{\mu}^j_{n\tau} \triangleq \tilde{\mu}_{n\tau}, \tag{2.29}$$

$$\mathrm{cov}_{\theta'}[Z_n | k, R] = \tilde{w}^{-1}_{n\tau} \sum_j \tilde{w}^j_{n\tau}\, \tilde{\Sigma}^j_{n\tau} \triangleq \tilde{\Sigma}_{n\tau}. \tag{2.30}$$

We can then write the expected outer product of each state-action pair as

$$\mathbb{E}_{\theta'}[Z_n Z_n^T | k, R] = \tilde{\Sigma}_{n\tau} + \tilde{\mu}_{n\tau}(\tilde{\mu}_{n\tau})^T. \tag{2.31}$$

Finally, letting $\tilde{\Sigma}_{n\tau}^{(X)}$ and $\tilde{\Sigma}_{n\tau}^{(A)}$ denote the state and action components of the covariance matrix respectively, and similarly defining $\tilde{\mu}_{n\tau}^{(X)}$ and $\tilde{\mu}_{n\tau}^{(A)}$, we can write the remaining expectation as

$$\mathbb{E}_{\theta'}[C^T C|k, R] = \mathrm{Tr}(K\tilde{\Sigma}_{n\tau}^{(X)}K^T + \tilde{\Sigma}_{n\tau}^{(A)}) + \|\tilde{\mu}_{n\tau}^{(A)} - K\tilde{\mu}_{n\tau}^{(X)} - m\|^2. \quad (2.32)$$

2.4 Experiments

2.4.1 Results on synthetic data

In this section we will empirically observe the behavior of EM on Gaussian MDPs of the form introduced in Section 2.3. These include

- *standard EM*, where the policy parameters are updated analytically by finding a fixed-point to Equations (2.26–2.28);
- *generalized EM*, where optimization is performed using the gradient $\nabla Q(\theta_{i-1}, \theta_{i-1})$ and a quasi-Newton method, LBFGS-B as stated earlier.
- *pseudo-gradient EM*, where a "gradient" as given by taking the steps provided by the standard EM approach and using this as the basis for a quasi-Newton method.

The simplest way to test these methods involves randomly generating the parameters of an MDP model and observing their convergence behavior. We generated transition parameters A_{ij} and B_{ij} from a standard uniform and components of the initial-state mean μ_0 uniformly in the range $[0, 5]$; covariance terms for these models were initialized diagonally with standard-deviations uniformly sampled in the range $(0, 5]$. The reward terms w_j and y_j were initialized similarly, M_j was the identity, and the "covariances" L_j were initialized using a random SPD matrix with eigenvectors uniformly distributed in $(0, 5]$.

As an aside, it is also worth noting that although we know the optimal policy will be deterministic, by allowing the exploration term σ to vary we obtain annealing-like behavior where local maxima are smoothed out

by a large initial value of σ. The plot in Figure 2.5 contains the average convergence behavior of the two EM variants on a series of 100 simple, 1-dimensional MDPs (a 3-dimensional parameter space). The trace of each optimization process is normalized to be in the range $[0, 1]$ and averaged across all models. The first thing to note is the poor performance of the standard EM algorithm as compared to the GEM algorithm. While these approaches will typically converge to the same local maximum, this behavior results from smaller step-sizes taken by the standard EM algorithm. This behavior stems from the high proportion of hidden data and is a situation that only worsens as the dimensionality increases.

We also contrast the EM-based approach to policy gradient methods including: (i) a gradient-free approach using *finite-differences* and common random numbers for variance reduction (i.e. PEGASUS [Ng and Jordan, 2000]); (ii) stochastic gradient ascent using the *vanilla policy-gradient* and the vanilla policy-gradient combined with the *optimal baseline*; (iii) the *natural actor-critic* [Peters and Schaal, 2008]. The convergence rates of these different algorithms can also be seen in Figure 2.5. Given a well chosen learning rate—and if the reward model induces a nice, broad surface with well-defined gradients—the policy-gradient methods perform quite well. It is worth noting, however, that these algorithms are greatly affected by the choice of learning rate. For most models it seemed possible to vary the learning rate of the policy-gradient methods in order to achieve performance comparable with GEM, but these learning rates did not generalize across multiple models and required multiple runs to obtain. One other tradeoff, however, is the time-complexity of these two classes of algorithms. Each iteration of the policy gradient algorithms runs in time $O(pk_{\max})$ where p is the number of trajectories, and $k_{\max}$ is the time-horizon; the EM-based algorithms are $O(k_{\max}^2)$. As a result, even accounting for the issue of learning rates, the fact that these algorithms are linear in $k_{\max}$ may make them seem more attractive for problems with a large time-horizon.

The differences in performance, however, are much greater when the reward function is rare, i.e. with support limited to only a small region of the state space. Figure 2.6 shows the convergence behavior of the algorithms

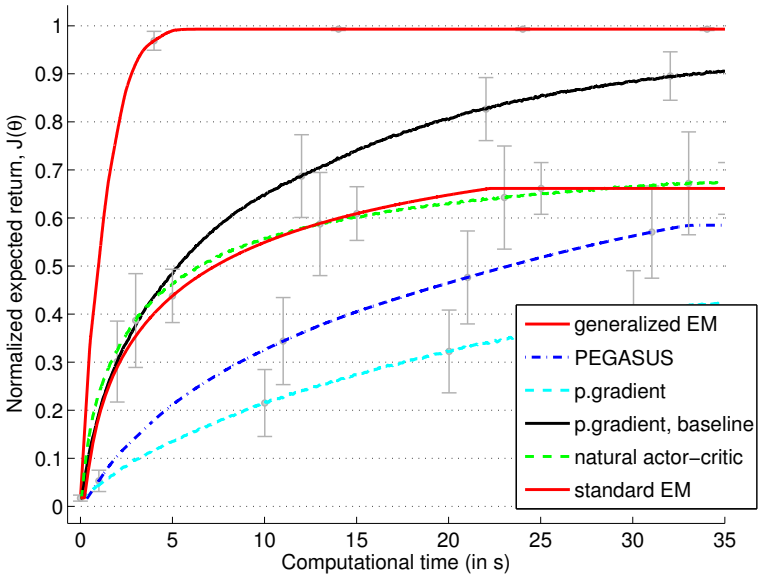

Figure 2.5: Convergence results for 100, randomly sampled 1-dimensional mixture of Gaussians MDPs; higher is better. Here the policy space is $\mathbb{R}^3$. For each algorithm we use the same tuning parameters (i.e. learning rate, number of trajectories, etc.) across different models. Also shown is the variation of this performance between different models. The two red-lines correspond to the EM algorithms proposed in this work, where the higher of these two lines corresponds to the GEM algorithm.

on a set of larger, 3-dimensional state- and action-space models (policies in these models are parameterized by $\theta \in \mathbb{R}^{13}$). In particular we see that the policy-gradient algorithms perform very poorly, and are for the most part unable to make any progress towards the goal. This is because in this space the reward model is much more rare than in lower dimensions, resulting in little-to-no gradient information except in a small region around the optimum and what little gradient exists is also likely washed out by the noise in the system. The GEM algorithm does not suffer from this problem not only because the iterations are analytic, but also because of the backward messages. In contrast, the policy gradient algorithms only perform a noisy forward pass, and so are much less likely to get good information about a

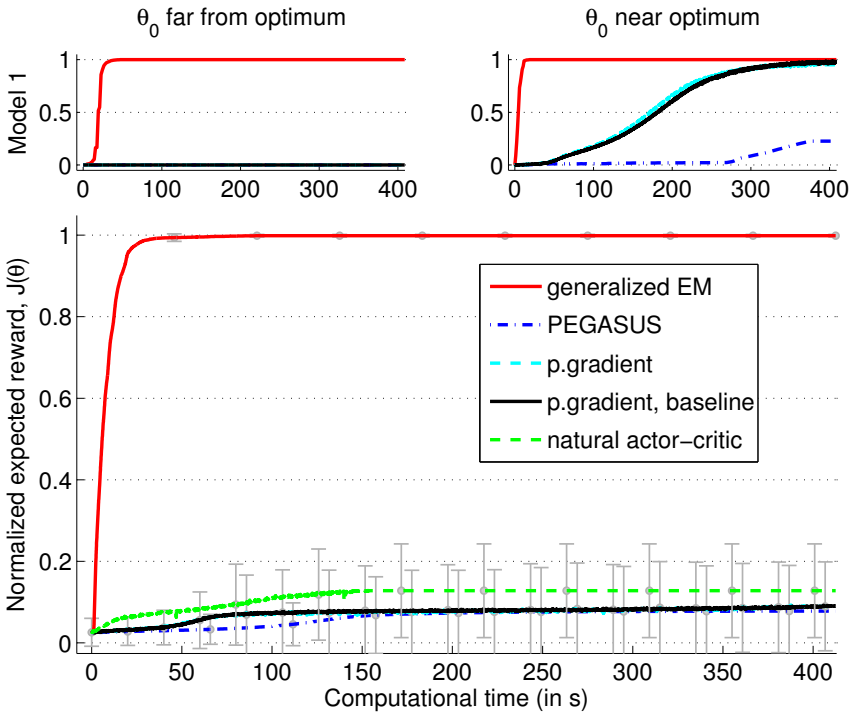

Figure 2.6: Convergence results for 20 randomly selected 3-dimensional mixture of Gaussians MDPs. Here the policy space is $\mathbb{R}^{13}$. The policy gradient algorithms did not have enough gradient information to make any progress in all but a few of the models. The two plots on top show the change in this behavior when the initial policy θ_0 is initialized closer to the optimum.

rare reward. We also experimented with starting the initial value of θ closer to the optimum, as can be seen from the top plots of Figure 2.6, and we see that the the policy gradients are able to make progress in this situation. This is not, however, an effective strategy: in order to find a closer value for θ_0 we were forced to initially solve the system using the GEM algorithm.

2.4.2 Robotic applications

Consider an n-jointed robotic system such that $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$ denote the joint angles, velocities, and accelerations respectively. Most such systems can be

described by the rigid-body dynamics

$$\ddot{q} = M^{-1}(q)(\tau - c(q, \dot{q}) - g(q)), \tag{2.33}$$

where $M(q)$ denotes the inertia matrix, $c(q, \dot{q})$ denotes the coriolis and centripetal forces, $g(q)$ is the force due to gravity, and τ the torques generated by the motors. The objective is then to control the evolution of the joints $[q; \dot{q}; \ddot{q}] \in \mathbb{R}^{3n}$ with actions given by some sequence of torques $\tau \in \mathbb{R}^n$. It is easy to see from (2.33), however, that the dynamics of this system are highly non-linear and as a result we cannot apply the techniques of Section 2.3. But the system can instead be reformulated in a different action-space that enforces linear dynamics.

A system can be called *feedback-linearizable* if there exists some function $\hat{\tau}(q, \dot{q}, \ddot{q})$ that cancels the natural dynamics of the system in some local neighborhood of $(q, \dot{q}, \ddot{q})$, where $\ddot{q}$ is some desired joint-space acceleration. That is, we want a function $\hat{\tau}$ that locally approximates the torque required to maintain some acceleration $\ddot{q}$ from the state $(q, \dot{q})$. In general this function can be obtained via an estimate of the inverse dynamics [Lewis et al., 2004]. Given the inverse dynamics, we can control the evolution of states $x = [q; \dot{q}]$ via actions $a = \ddot{q}$ where the dynamics are linear.

While it is often possible to linearize the dynamics of such a system, this can drastically change the reward model necessary to induce some desired behavior. It is also frequently the case that the reward model depends on non-linear terms such as the end-effector position $\hat{x}(q)$, typically written via the forward-kinematics of the system and depending on parameters such as the link lengths and masses [again, see Lewis et al., 2004]. Although the reward may be a simple quadratic with respect to the end-effector position, this will in general be non-linear in x and a. These are precisely the situations where the mixture of Gaussians approach presented earlier should prove useful. On one hand this formulation immediately allows us to tackle multimodal regulation tasks whose rewards can be specified in the form of Equation (2.20). Another possibility, though, would be to fit the model either to some known functional form or to data.

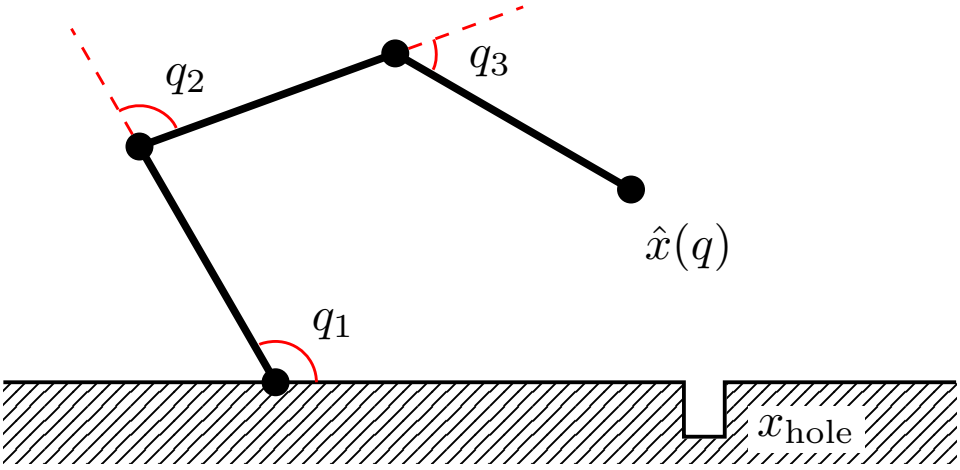

Figure 2.7: Model of a robot arm "peg-in-hole" task.

One simple application of this technique on a robotic model is the "peg-in-hole" task, a depiction of which is shown in Figure 2.7. The goal of this task is to move the end-effector into some position x_{hole} and regulate about this point. We can specify the reward as

$$r(x,u) = r(q,\dot{q},\ddot{q}) = \exp\left\{-\lambda_1\|\hat{x}(q) - x_{\text{hole}}\|^2 - \lambda_2\|\dot{q}\|^2 - \lambda_3\|\ddot{q}\|^2\right\}.$$

Rather than attempting to fit this entire model, we can instead restrict ourselves to the reward-terms depend on q. In this paper we fit the model using the sparse, pseudo-input Gaussian Processes regression of [Snelson and Ghahramani, 2006]. The regression process results in a linear-combination of Gaussians with parameters (w_j, m_j, S_j) that can be used to approximate the full reward model:

$$\begin{aligned} r(q,\dot{q},\ddot{q}) &\propto \sum_j w_j \mathcal{N}(q; m_j, S_j) \cdot \exp\left\{-\lambda_2\|\dot{q}\|^2 - \lambda_3\|\ddot{q}\|^2\right\} \\ &\propto \sum_j w_j \mathcal{N}([q,\dot{q},\ddot{q}]^T; y_j, L_j), \end{aligned}$$

for $y_j = [m_j, 0, \ldots, 0]^T$ and $L_j = \text{blkdiag}(S_j, \lambda_2^{-1}I, \lambda_3^{-1}I)$. Given this information we can now perform policy search over the 22-dimensional space of

all policies. Here we use a maximum time-horizon $k_{\max} = 100$. Figure 2.8 shows a trace of the resulting optimization process; the resulting policy is successfully able to move the arm's end-effector to the position x_{hole} and from there regulates about this point. Here, although not a simple problem, the reward is relatively broad, and the natural actor-critic was able to do well. Due to the analytic nature of the GEM approach, however, we are still able to improve on this policy in the long-run. The GEM algorithm actually converges quite quickly, but each iteration is adversely affected by the $O(k_{\max}^2)$ time complexity. Based on these results we also experimented with putting a single, very peaked reward at the resulting target joint angles. These results are also shown in Figure 2.8, and we can see that the analytic GEM algorithm greatly outperforms the natural actor-critic with rare rewards.

2.5 An extension to semi-Markov Decision Processes

Formally we can define an SMDP as a continuous-time controlled stochastic process $Z(t) = (X(t), A(t))$ consisting, respectively, of states and actions at every point in time t. In particular we will assume that the system transitions at random *arrival times* T_n and that the process is constant in between jumps, i.e. $Z(t) = Z_n = (X_n, A_n)$ for all $t \in [T_n, T_{n+1})$. It is this use of random transition times that makes this a *semi-Markov* Decision Process (SMDP). In order to handle this generalization we will introduce random *sojourn times* $S_n > 0$ representing the amount of time spent in the nth state, i.e. $T_{n+1} = T_n + S_n$. In order to fully specify the SMDP model we repeat the standard MDP models defined over the extended state space $\mathcal{Z} = \mathcal{X} \times \mathcal{A}$, i.e.

- a transition model $f_\theta(z_{n+1}|z_n)$, initial-state model $\mu_\theta(z_1)$, and reward model $r(z_n)$;

and in addition we must specify

- a time model $g(s_n|z_n)$;

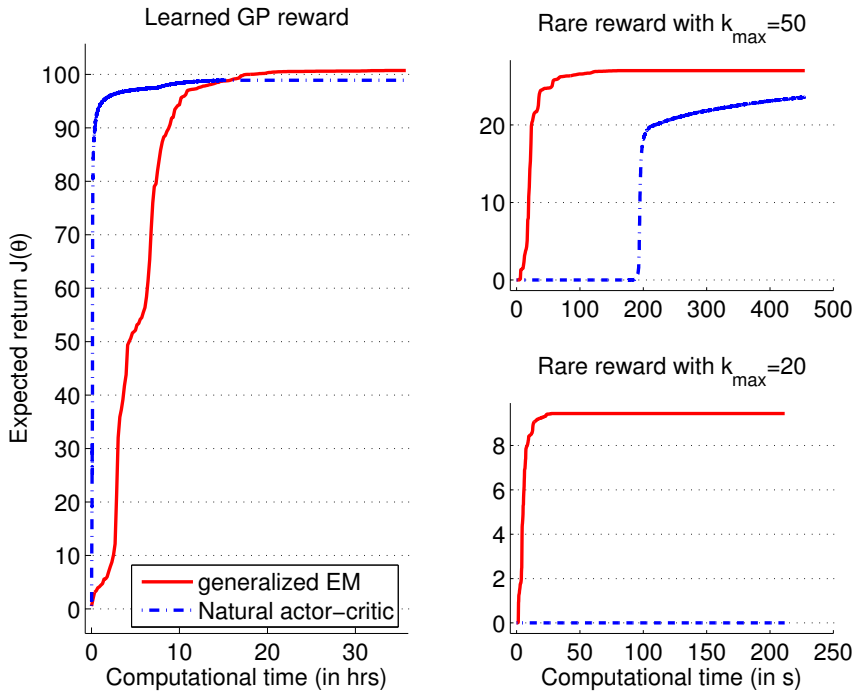

Figure 2.8: A trace of the optimization process for the 3-jointed robot arm model using the learned reward (left) and a rare reward (right).

- and a discount rate $\beta > 0$, analogous to the standard discount factor.

Importantly, the time model does not depend on the duration of the previous interval. See Figure 2.9 for an illustration of this process.

We can now write the joint probability over sequences of state-action pairs and sojourn times as

$$p_\theta(z_{1:k}, s_{1:k}) = \mu_\theta(z_1)\, g(s_1|z_1) \prod_{n=2}^{k} g(s_n|z_n)\, f_\theta(z_n|z_{n-1}) \tag{2.34}$$

for any choice of horizon k. Unlike in an MDP, however, our discounting behaves differently in order to take into account the variable time in each state. In particular, we will discount the reward continuously over our entire trajectory using a parameter β, which is a continuous-time analogue of the

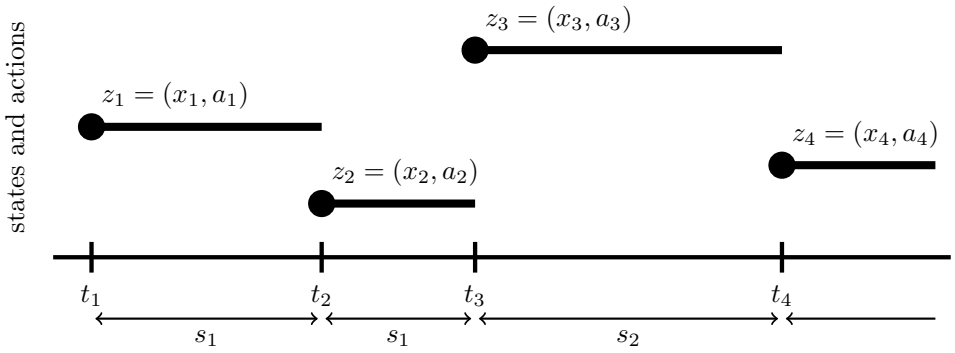

Figure 2.9: Relationship between arrival times t_n, sojourn times s_n, and the system state $z_n = (x_n, a_n)$ in the SMDP model.

discount factor γ used in previous sections. We can then write the objective function as

$$J(\theta) = \mathbb{E}_\theta \left[\int_0^\infty e^{-\beta t} \, r(Z(t)) \, dt \right]$$

which because of the jump-Markov nature of our transitions will simplify to

$$= \mathbb{E}_\theta \left[\sum_{n=1}^{\infty} \beta^{-1} e^{-\beta T_n} (1 - e^{-\beta S_n}) \, r(Z_n) \right]. \tag{2.35}$$

As a result, we can also think of this as an MDP with discrete steps, but where the discounting for each step is stochastic, and more importantly non-uniform. Intuitively this means that we have a system where we more severely discount those steps that take longer.

Based on the intuition developed earlier for MDPs we can interpret the discount terms in (2.35) as a distribution over random time horizons k and write the following joint distribution over paths and path lengths:

$$p_\theta(k, z_{1:k}, s_{1:k}) = e^{-\beta t_k} (1 - e^{-\beta s_k}) \, p_\theta(z_{1:k}, s_{1:k}). \tag{2.36}$$

This distribution is again defined over a trans-dimensional space $\bigcup_{k=1}^{\infty}(\{k\} \times \mathcal{Z}^k \times \mathbb{R}_+^k)$ where $\mathbb{R}_+$ is the set of strictly positive real numbers. Also, note that $t_k = \sum_{n=1}^{k-1} s_n$ is defined as the arrival time of the kth step and as a result is a deterministic function of the sojourn times up to step k. Unlike in the standard MDP case, however, this distribution is not nearly as straightforward to work with. Nor is there the simple division between path-lengths and paths.

Proposition 2. *The joint distribution introduced in* (2.36) *is properly defined and normalized, i.e. it integrates to 1.*

Proof. As noted above, the density defined in (2.36) is with respect to a trans-dimensional measure defined over the space $\bigcup_{k=1}^{\infty}(\{k\} \times \mathcal{Z}^k \times \mathbb{R}_+^k)$. We can first easily see that conditioned on k we have a path distribution just as in previous sections defined over $\mathcal{Z}^k \times \mathbb{R}_+^k$, which is well-defined so long as the underlying transition and time densities are well defined. As long as the distribution then gives rise to a proper marginal in K, then the joint density is proper.

For any given k we can integrate out the path components, allowing us to write the marginal as

$$p_\theta(k) = \mathbb{E}[e^{-\beta T_k}] - \mathbb{E}[e^{-\beta T_{k+1}}]$$

where again, $T_k = S_1 + \cdots + S_{k-1}$ is the starting time of the kth step, or equivalently the ending time of the $(k-1)$th step. From here we can sum over the values of k, arriving at

$$\sum_{k=1}^{\infty} p_\theta(k) = \sum_{k=1}^{\infty}\Big(\mathbb{E}[e^{-\beta T_k}] - \mathbb{E}[e^{-\beta T_{k+1}}]\Big),$$

i.e. a telescoping series involving only the first and last terms for finite k, which can be written as

$$\begin{aligned} &= \mathbb{E}[e^{-\beta T_1}] - \lim_{k\to\infty} \mathbb{E}[e^{-\beta T_k}] \\ &= \mathbb{E}[e^{-\beta T_1}] = 1. \end{aligned}$$

The final equality holds since T_k is monotonically increasing due to the requirement that each $S_n > 0$ and $T_1 = 0$. □

With this path density in place we can now utilize a reward likelihood $p(R|k, z_k) = r(z_k)$ as in previous sections. This allows us to write the following joint density

$$\begin{aligned} p_\theta(k, z_{1:k}, s_{1:k}, R) &= p_\theta(k, z_{1:k}, s_{1:k}, R) \\ &= p_\theta(k, z_{1:k}, s_{1:k})\, p(R|k, z_k) \end{aligned}$$

corresponding to the complete data likelihood. We can then marginalize this density over the hidden data or condition on the observed data to write the incomplete data likelihood and posterior distribution respectively as

$$\begin{aligned} p_\theta(R) &= \sum_{k=1}^{\infty} \iint p_\theta(k, z_{1:k}, s_{1:k}, R)\, ds_{1:k}\, dz_{1:k}, \\ p_\theta(k, z_{1:k}, s_{1:k}|R) &= \frac{p_\theta(k, z_{1:k}, s_{1:k}, R)}{p_\theta(R)}. \end{aligned}$$

We can then easily show that maximizing the incomplete data likelihood is equivalent to maximizing the expected reward of the SMDP.

Proposition 3. *The objective function $J(\theta)$ from* (2.35) *is proportional to the marginal likelihood defined in* (2.4)*. More precisely,*

$$J(\theta) = \beta^{-1}\, p_\theta(R).$$

Proof. Given the joint distribution $p_\theta(k, z_{1:k}, s_{1:k})$ we can rewrite our objective from (2.35) as

$$\begin{aligned} \beta\, J(\theta) &= \mathbb{E}_\theta\left[\sum_{n=1}^{\infty} e^{-\beta T_n}(1 - e^{-\beta S_n})\, r(Z_n)\right] \\ &= \sum_{k=1}^{\infty} \iint p_\theta(z_{1:k}, s_{1:k})\, e^{-\beta t_k}(1 - e^{-\beta s_k})\, r(z_k)\, ds_{1:k}\, dz_{1:k} \end{aligned}$$

$$= \sum_{k=1}^{\infty} \iint p_\theta(k, z_{1:k}, s_{1:k})\, r(z_k)\, ds_{1:k}\, dz_{1:k}$$
$$= p_\theta(R). \qquad \square$$

Similar to the MDP case we have obtained this result by exchanging the order of integration and summation and pulling the discount factor into the previously introduced distribution from (2.36).

At this point we can also note how the formulations of this section simplify in the MDP case, and more importantly why these simplifications do not hold for SMDPs. In particular, when the sojourn times are given deterministically by $S_n = 1$ we recover a standard infinite-horizon, discounted MDP with a discount factor of $\gamma = e^{-\beta}$. It is precisely because the sojourn times are independent of the states Z_n that we can factorize the joint distribution in (2.2) as a prior over path lengths K and paths conditioned on this length $Z_{1:K}$. Unfortunately this interpretation does not hold in the case of more general SMDPs. By looking at the discount factors in (2.36) we can see that the probability of a specific trajectory length K is a function of all sojourn times $S_{1:K}$, and as a result the distribution over the random variable K depends on an infinite number of sojourn times. However, while the SMDP formalism does not have as clean of a probabilistic interpretation as MDPs, we can still apply this model by working directly with the joint distribution.

Finally, given these quantity we can focus on the problem of maximizing the incomplete data likelihood. Again, we will utilize an EM algorithm in order to do so, and can write the Q function as

$$\begin{aligned} Q(\theta, \theta') &= \mathbb{E}_{\theta'}\left[p_\theta(K, Z_{1:K}, S_{1:K}, R) \middle| R\right] \\ &= \sum_{k=0}^{\infty} \sum_{n=0}^{k} \int p_{\theta'}(k, z_n | R)\, \log \pi_\theta(z_n)\, dz_n + \text{const.} \end{aligned} \tag{2.37}$$

Here we have integrated out the sojourn time variables $S_{1:K}$, but these still affect the posterior distribution due to their effect on the path length K. Note also that this is very similar to the standard MDP formulation, however

as noted earlier we can no longer factorize the joint $p_{\theta'}(k, z_n)$. In the next subsection we will detail the E-step necessary to construct this distribution; the M-step follows directly from the exposition in Section 2.2.2

2.5.1 The E-step

We start by writing the marginal distribution as the integral of the posterior distribution with respect to all those terms other than k and z_n,

$$p_\theta(k, z_n|R) = \int p_\theta(k, z_{1:k}, s_{1:k}|R)\, dz_{1:n-1}\, dz_{n+1:k}\, ds_{1:k}$$

which can then be broken into those components that come before and after n respectively, i.e.

$$\begin{aligned} \propto \int e^{-\beta t_n}\, p_\theta(z_{1:n}, s_{1:n-1})\, dz_{1:n-1}\, ds_{1:n-1} \quad \times \\ \int e^{-\beta(t_k - t_n)}(1 - e^{-\beta s_k})\, r(z_k)\, p_\theta(z_{n+1:k}, s_{n:k}|z_n)\, dz_{n+1:k}\, ds_{n:k}. \end{aligned} \tag{2.38}$$

Recall that the arrival times are given by the sum of previous sojourn times $t_n = s_1 + \cdots + s_{n-1}$, and note that we have also omitted the constant of proportionality, given by $p_\theta(R|k)$. We should emphasize the fact that we are *not* integrating over z_n, which enables us to break the original integral into two independent integrals.

Following the notation from Section 2.2.1 we can now introduce the forward and backward messages, $\alpha_n(z_n)$ and $\beta_\tau(z_n)$ respectively[2], where again $\tau = k - n$. In particular these messages are given exactly by the two components of (2.38). Just as with the standard MDP we can then write the marginal as the product of forward and backward messages

$$p_\theta(k, z_n|R) \propto \alpha_\theta(z_n)\, \beta_{k-n}(z_n). \tag{2.39}$$

[2]The notation here is slightly confusing in that we have a term β denoting the continuous discount factor and $\beta_\theta(\cdot|\tau)$ denoting the backward messages. This confusion, however, seems unavoidable as both of these terms are unanimously used in their respective literatures. To somewhat alleviate this confusion we note that the backward messages are *always* subscripted.

It is crucial to note, however, that unlike in the earlier MDP formulation these messages are not probability distributions due to the way the discount factors have been split between the forward and backward components, namely:

$$\underbrace{e^{-\beta(s_0+\cdots+s_{n-1})}}_{\text{forward}}\underbrace{e^{-\beta(s_n+\cdots+s_{k-1})}(1-e^{-\beta s_k})}_{\text{backward}}.$$

This causes no technical (or conceptual) difficulties, though, because when combined in (2.39) these messages form the desired probability distribution. This is similar in spirit to techniques used to maintain numerical stability when working with hidden Markov models [Bishop, 2006].

Finally, by integrating the components of (2.38) successively we can recursively define the messages as

$$\alpha_n(z_n) = \int \alpha_\theta(z_{n-1})\, f_\theta(z_n|z_{n-1})\, dz_{n-1} \times \int e^{-\beta s_{n-1}}\, g(s_{n-1}|z_{n-1})\, ds_{n-1}, \tag{2.40}$$

$$\beta_\tau(z_n) = \int \beta_{\tau-1}(z_{n+1})\, p_\theta(z_{n+1}|z_n)\, dz_{n+1} \times \int e^{-\beta s_n}\, g(s_n|z_n)\, ds_n. \tag{2.41}$$

Here we can see that we have the standard MDP forward message recursions multiplied by an additional integral due to the sojourn times. Given the format of these two messages we can further introduce what we call an "expected discount factor"

$$\gamma(z) = \int e^{-\beta s}\, g(s|z)\, ds \tag{2.42}$$

which corresponds to the integral over sojourn times noted above. We can consider this term as a generalization of the MDP formalism wherein discount factors are no longer constant and instead depend on the current state and the action taken from that state. Further, we can see that for any exponential-family distribution this integral will exist in closed form.

2.5.2 Discrete models with Gamma-distributed time

The methods presented in previous sections, as with all EM-based procedures, provide a "meta-algorithm" which depends upon the exact models in use. In this section we present a simple model for the purposes of illustrating the procedure, namely a model where the states and actions are discrete and sojourn times are given by a Gamma distribution. While simple, this model nonetheless presents an interesting scenario for planning and control domains because it can naturally be extended to cases when we want to reason about more complex distributions over the time to complete an action. In our experiments, we define the following standard discrete models:

$$\begin{aligned} \mu(x) &= \mu_x, \\ f(x'|x,a) &= P_{xax'}, \\ r(x,a) &= R_{xa}, \\ \pi_\theta(a|x) &= \theta_{xa}. \end{aligned}$$

Finally we will also assume sojourn times are Gamma distributed random variables with density,

$$g(s|x,a) = \Gamma(s; k_{xa}, \sigma_{xa}).$$

In this setting the expected discount factor noted in (2.42) can be written as the matrix

$$\begin{aligned} \gamma_{xu} &= \int \Gamma(s; k_{xu}, \sigma_{xu})\, e^{-\beta s}\, ds \\ &= \int s^{k_{xu}-1} \frac{\exp\big(-(\beta + \sigma_{xu}^{-1})\, s\big)}{\Gamma(k_{xu})\, \sigma_{xu}^{k_{xu}}}\, ds = (1+\beta\sigma_{xu})^{-k_{xu}}. \end{aligned}$$

This particular form arises purely from the use of Gamma-distributed sojourn times, and in fact we can imagine extending this to continuous spaces using functions $k(x,u)$ and $\sigma(x,u)$.

Under this formulation we will let β_{xu}^{τ} denote the τ-step backward message and α_x^n the n-step forward message in state-space. We can easily see

that the extended-state space forward message is given by $\alpha_x^n \, \theta_{xa}$. Given the specific models introduced earlier we can explicitly write these messages as

$$\alpha_x^n = \sum_{x',a'} \alpha_{x'}^{n-1} \, \theta_{x'a'} \, P_{x'a'x} \, \gamma_{x'a'}, \tag{2.43}$$

$$\beta_{xa}^\tau = \gamma_{xa} \sum_{x',a'} \beta_{x'a'}^{\tau-1} \, P_{xux'} \, \theta_{x'a'}, \tag{2.44}$$

where the messages are initialized with $\alpha_x^1 = \mu_x$ and $\beta_{xa}^0 = R_{xa}(1 - \gamma_{xa})$. By plugging these terms into the Q-function defined in (2.37) we can write

$$\begin{aligned} Q(\theta, \theta') &\propto \sum_{k=0}^{\infty} \sum_{n=0}^{k} \sum_{u,x} (\log \theta_{xu}) \, \alpha_x^n \, \theta'_{xu} \, \beta_{xu}^{k-n} \\ &= \sum_{u,x} (\log \theta_{xu}) \, \theta'_{xu} \left[\sum_{n=0}^{\infty} \alpha_x^n \right] \left[\sum_{\tau=0}^{\infty} \beta_{xu}^\tau \right], \end{aligned}$$

where the second equality can be obtained by rearranging the sums over k and n. This alternate formulation is particularly useful in discrete models where the sum over forward and backward messages can expressed as finite quantities, i.e. a vector and a matrix respectively. Further, given this formulation we can optimize the Q-function for each state x individually, which is possible because in discrete domains we can find the optimal action to take for each state regardless of the probability of visiting that state. By taking the gradient of the log-policy $\nabla \log \pi_\theta(a|x) = \theta_{xa}^{-1}$ and solving $\nabla Q(\theta, \theta') = 0$ for θ, subject to the constraint that $\sum_u \theta_{xu} = 1$ for each x, we arrive at the following solution methods:

$$\theta_{xu} \propto \theta'_{xu} \sum_{\tau=0}^{\infty} \beta_{xu}^\tau, \tag{EM}$$

$$\theta_{xu} = \delta_{m(x)}(u) \quad \text{where } m(x) = \arg\max_{u'} \textstyle\sum_{\tau=0}^{\infty} \beta_{xu'}^\tau. \tag{greedy-EM}$$

Here the EM solution is performing exactly the optimization described above, while the greedy-EM solution, however, myopically chooses for each state x the one action u that maximizes the total future rewards when taken

from that state. In particular, the greedy solution can be seen as iterations which correspond to repeated M-steps which skip intermediate E-steps as is shown by [Toussaint et al., 2006], and in this sense this method is equivalent (again only for discrete models) to policy iteration. In larger discrete models, however, the EM approach has the advantage over policy iteration in that it is possible to prune computation (by using the forward messages α) in a principled manner; see [Toussaint et al., 2006] for more details.

We first test these algorithms on a small, 16-state, 5-action domain with randomly generated transition and sojourn parameters, as well as a randomly generated reward model. The principal challenge of this domain, over other discrete domains, is to take advantage of the structure in the sojourn time distribution. The top-left plot of Figure 2.10 displays convergence properties of the described algorithms as well as a comparison to a standard policy-gradient method [see e.g. Baxter and Bartlett, 2001]. In particular we should note that the resulting model was densely connected which allows for quick travel across the space, and explains the very good performance of the stochastic policy gradient algorithm. Also shown for policy gradients are error-bars corresponding to one standard deviation. The other methods don't need error bars because they are deterministic.

Building upon these results, the top-right plot shows the algorithms learning in a more structured environment. In particular the model used has grid-structured transitions on a small 4-by-4 grid. This model is especially interesting because we specify different Gamma-distributions for the inner nodes than the outer nodes such that the inner nodes move much more slowly. Also, we use a sparse reward model where most states have negligible reward and one state has high reward. The most important thing to note from this sub-figure is that the policy gradient method starts to break down under this sparse transition and reward model, even though the size of the state and action spaces are the same as in the previous example.

Lastly the bottom-left plot of this figure displays the progress of these algorithms on a much larger 20-by-20 grid, i.e. one in which there are 2000 state-action pairs. Similar to the previous example there is a single (relatively) large reward in the upper right corner of the grid and inner nodes

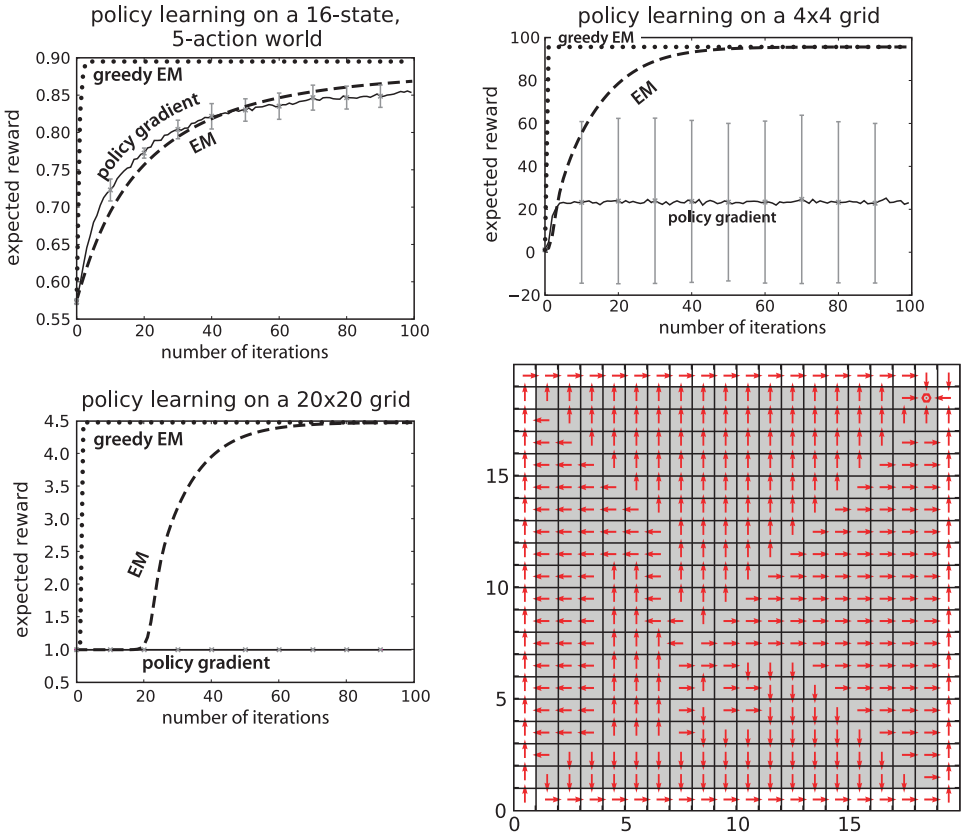

Figure 2.10: Results on various discrete SMDP models. The top-left plot shows convergence of the described algorithms on a randomly generated, dense model; the top-right plot shows performance on a model of the same size but with grid-like transitions and more structured transition times. The bottom-left plot shows performance on a larger grid domain. The bottom-right shows the learned policy in the larger domain. For both grid domains there is a single large reward (denoted with a dot) and transitions taken outside of the edge states (denoted with grey) have a larger expected sojourn time.

with much slower sojourn times. Here we see that the EM algorithms vastly out-perform the policy gradient method and the learned policy successfully skirts the outside edge of the state-space in order to most quickly get to the high reward. Here the policy gradient method has relatively low-variance because it is not able to make any progress (i.e. it is stuck exploring a plateau with very little gradient information).

2.6 Chapter summary and conclusions

In this chapter we presented an alternative view of model-based policy search as one of maximum likelihood in an an equivalent probabilistic model. We then used this procedure to develop a novel policy search method based on linear-Gaussian transitions and mixture-of-Gaussians rewards. This approach is able to fit any arbitrary smooth reward function—essentially extending the approach of LQR to arbitrary rewards. We showed results of this procedure on randomly generated MDPs, as well as an example robotic application. These results show that for rare rewards the analytic policy search methods developed in this chapter are able to out-perform simpler, simulation-based policy gradient method. Further, the analytic nature of this search procedure allows for broad initial policies which aid in the exploratory behavior of early iterations. One of the primary benefits of this probabilistic perspective on decision problems is that it provides a standard framework for integrating approaches from more standard machine learning algorithms, such as those used for supervised learning and inference. As an example, the derivations the mixture-of-Gaussians model were simplified by their analogy to work done in hidden Markov models and linear dynamical systems.

Later in this chapter we also extended the probabilistic approach to that of semi-Markov models. Although this is presented as a proof-of-concept, our work was the first to extend the maximum-likelihood method to this type of decision problem. This presents an interesting area of future research as it provides for a research path for extending these ideas to hierarchical planning and RL, such as that discussed in [Ghavamzadeh and Mahadevan,

2007]. This work does, however, directly allow for continuous time models to be solved using EM.

Further, in this chapter while we presented an EM algorithms for solving MDPs, solving one could also consider more general variational approaches to attacking these decision problems. To some degree this has been proposed by [Furmston and Barber, 2010], however more specific approaches to take advantage of specific model structure would substantially push these methods forward. Finally, as noted in later work by [Furmston and Barber, 2011] when building on our earlier mixture-of-Gaussians model, when the parameter updates are linear in the necessary expectations, such as $\mathbb{E}[Z_n|k]$, the computation of the gradient can be sped up to linear complexity. This observation may allow for quite general "exponential-family MDPs" in a similarly vein to their application to many standard inference tasks; see e.g. [Wainwright and Jordan, 2008].

Chapter 3

Bayesian methods for solving Markov Decision Processes

In Chapter 2 we introduced a probabilistic model over latent paths $(K, Z_{1:K})$ and a dummy observation R such that the marginal likelihood of this observation under parameters θ coincides with the expected reward of those parameters. In this chapter we will utilize a similar probabilistic model, except rather than optimizing the policy parameters we will perform a full Bayesian analysis of these parameters, producing a posterior $p(\theta|R)$ that is proportional to the expected reward. The contributions of this chapter are mostly algorithmic, in that we present the first approach and application of Markov Chain Monte Carlo (MCMC) to the problem of policy search. However, we will also show that this approach has benefits for gradient-free policy search that is useful in situations where the gradient information is very poor.

We will first give a brief overview explaining the extension of the previous Chapter's approach to the Bayesian setting in Section 3.1. We will then provide a general, relatively high-level introduction to Markov Chain Monte Carlo (MCMC) in Section 3.2; we will however have to bring in some technical details in order to deal with the trans-dimensional nature of $(K, Z_{1:K})$. In Section 3.3 we will introduce a novel Bayesian approach to policy search using MCMC and in Section 3.4 we will introduce a number of improve-

ments to this basic algorithm in order to improve the mixing properties of the Markov chain. We will also show in Section 3.5 how to modify this MCMC approach to optimize the policy parameters. Finally in Section 3.6 we will provide a number of experiments testing the performance of this procedure and the affect of the improved sampling process.

3.1 A Bayesian interpretation of the MDP problem

In this chapter we return to the problem of solving Markov decision processes with parameterized policies. In particular, we will treat the vector of unknown policy parameters θ as a latent random variable. Conditioned on this random variable we write the posterior from (2.8) as

$$p(k, z_{1:k}|R, \theta) \propto p(k, z_{1:k}|\theta)\, p(R|k, z_k)$$

This term is exactly the same as in Chapter 2, we have just rewritten it in such a way as to emphasize the fact that θ is now a random variable. With this notational detail out of the way we will now introduce a prior over the parameter vector $p(\theta)$ and write the joint posterior over paths and parameters as

$$p(\theta, k, z_{1:k}|R) \propto p(\theta)\, p(k, z_{1:k}|R, \theta). \tag{3.1}$$

By marginalizing this quantity with respect to the path parameters $(k, z_{1:k})$ we can easily see that we obtain

$$p(\theta|R) \propto p(\theta)\, p(R|\theta). \tag{3.2}$$

In other words, ignoring the prior term $p(\theta)$ we can see that the marginal posterior for policy parameters is proportional to the expected reward of those parameters (via Proposition 1). By applying the results of the previous chapter we could obtain the *maximum a posteriori* (MAP) estimate by directly optimizing this quantity.

Instead, in this chapter we will introduce a method which provides a sample-based, Monte Carlo approximation to the posterior. By construction, samples from the marginal posterior (3.2) will concentrate in regions of high expected reward $J(\theta)$. This is a particularly interesting feature in situations where the reward function is concentrated in a region of low prior probability $p(k, z_{1:k}|\theta)$, which is often the case in high-dimensional control settings. Note that if we wanted to estimate $J(\theta)$ using importance sampling, then the distribution given by $p(k, z_{1:k}|R, \theta)$ corresponds to the optimal zero-variance importance distribution.

Finally, as noted above, the posterior is proportional to the expected reward only when we ignore the effect of the prior distribution. However, if the integral $\int J(\theta)\, d\theta$ is finite, one possible option is an improper, uninformative prior $p(\theta) \propto 1$. Another reasonable choice is a uniform distribution on a bounded region of the parameter space. This will restrict samples to a fixed region but will not alter the expected reward surface inside that region. Alternatively we can abandon uninformative priors and treat this term as a regularizer which limits the policy complexity. For example we could use either Gaussian or Laplace priors centered at zero, corresponding to the standard use of ℓ_2 or ℓ_1 penalties respectively. The so called "spike and slab" prior is another sparsity-inducing prior, like the ℓ_1 penalty, often used in Bayesian analysis [Mitchell and Beauchamp, 1988]. In this work we will focus on uninformative priors for the sake of simplicity, although we note that a broad Gaussian is a good compromise, corresponding to an ℓ_2 regularizer with a small penalty parameter. We will return to the problem of sparsity-inducing priors in Chapter 4.

3.2 Markov Chain Monte Carlo

Although a complete description of Markov Chain Monte Carlo (MCMC) is beyond the scope of this work, in this section we will provide a brief introduction—specifically introducing the ideas and notation we will need in order to apply this technique to the problem of policy search. We will largely follow the exposition of [Green, 2003]; see also the excellent introduc-

tions of [Geyer, 2010] or [Robert et al., 1999, chapter 7] for a more in-depth treatment. The algorithm that we will specifically describe in this section is known as reversible jump MCMC as introduced by [Green, 1995], or the "Metropolis-Hastings-Green" algorithm as it is referred to by Geyer. This procedure is a generalization of the widely celebrated Metropolis-Hastings algorithm, but as we will see, this extension allows for the use of much more general state spaces. In particular it is often used for sampling the parameters of mixture models where the number of components is unknown [Richardson and Green, 1997]—exactly the trans-dimensional setting that we are confronted with.

We will now consider the problem of simulating a distribution η defined over some general state space $\mathcal{X}$. Rather than directly sampling from the distribution of interest, which may not even be possible, an MCMC method produces an ergodic Markov chain $\{X^{(i)}\}$ whose stationary distribution is η. In order to ensure the Markov chain *targets* the proper stationary distribution we will introduce a transition kernel K which satisfies the detailed balance condition

$$\iint_{A,B} \eta(dx)\, K(x, dx') = \iint_{A,B} \eta(dx')\, K(x', dx) \tag{3.3}$$

for all Borel sets $A, B \subset \mathcal{X}$. Transitions under this kernel are defined by some proposal (or instrumental) measure $q(x, dx')$ from which we sample a candidate state x' and accept this state with probability $\alpha(x, x')$, which we will derive shortly. If the candidate state is rejected the Markov chain will remain in state x. We can now more explicitly write the kernel as

$$K(x, dx') = \alpha(x, x')\, q(x, dx') + (1 - r(x))\, \delta_x(dx'), \tag{3.4}$$

where $r(x) = \int \alpha(x, x')\, q(x, dx')$ is the total probability of accepting a move away from x and $\delta_x(dx')$ denotes a Dirac mass at x. Plugging this kernel into (3.3) we can note that due to the rejection step there exists an atom at x resulting in the quantity $\int_{A\cap B} \eta(dx)\,(1 - r(x))$ on both sides of the equality. By subtracting this from each side, we can rewrite the detailed

balance condition as

$$\iint_{A,B} \eta(dx)\, q(x, dx')\, \alpha(x, x') = \iint_{A,B} \eta(dx')\, q(x', dx)\, \alpha(x', x). \tag{3.5}$$

It can be shown [Green, 1995] that the joint measure $\eta(dx)\, q(x, dx')$ is dominated by a symmetric measure λ on $\mathcal{X} \times \mathcal{X}$. Letting g be the density (Radon-Nikodym derivative) of the joint measure with respect to the dominating measure we can rewrite the detailed balance condition one last time as

$$\iint_{A,B} \alpha(x, x')\, g(x, x')\, \lambda(dx, dx') = \iint_{A,B} \alpha(x', x)\, g(x', x)\, \lambda(dx', dx). \tag{3.6}$$

From here we can see that an appropriate choice for the acceptance probability is as follows:

$$\alpha(x, x') = \min\left\{1, \frac{g(x', x)}{g(x, x')}\right\}. \tag{3.7}$$

If both the proposal kernel and target distribution have densities with respect to the same base measure, which we will also denote with $\eta(\cdot)$ and $q(\cdot|x)$, we can write this as the standard Metropolis-Hastings (MH) acceptance probability

$$\alpha_{\mathrm{MH}}(x, x') = \min\left\{1, \frac{\eta(x')}{\eta(x)} \cdot \frac{q(x|x')}{q(x'|x)}\right\}. \tag{3.8}$$

The MH approach is still quite general as long as proposal densities can be defined with respect to the same base measure as the target distribution. This, for example, allows for a mixture of proposal kernels where the mixing probability is state independent. The Gibbs sampler is a further special case of this mechanism, where one chooses to individually update components with proposals $q(x_j|x'_{-j})$ given exactly by the marginal densities of x_j; here the subscript in x'_{-j} is used to denote all variables other than the jth. In this case the proposal densities will cancel with the target densities and we will be left with a acceptance probability of 1.

We will see, however, a need for the full generality of (3.7) in situations

where the the proposals are singular with respect to the equilibrium distribution. As is the case in this work, this is often necessary for state spaces which are the union of sets of varying dimensions.

One last point that becomes obvious from the acceptance ratios is that it is only necessary to evaluate the required densities up to a constant of proportionality. We can easily see this in the case of the standard Metropolis-Hastings move (3.8), as any constant associated with the density over $\eta(x')$ will cancel due to the presence of $\eta(x)$; similarly for the proposal distribution. This is crucial to the practicality of these methods, as the normalizing constants are often intractable to compute—in fact in the case of policy search this quantity is obtained by integrating $J(\theta)$ over the space of policy parameters. For the more general ratio in (3.7) we can see that a similar situation also holds, where typically the joint density f will involve a ratio of posterior densities and allow us again to ignore the intractable constant.

3.3 Reversible jump MCMC for Bayesian policy search

We will now move on to the central problem of interest for this chapter, namely sampling from the marginal distribution $p(\theta|R)$. As noted earlier, samples generated from this distribution will concentrate in regions of high expected reward. In general, however, it is not possible to directly simulate from this posterior due both to the latent variables and the effect of conditioning on R. Instead, in this section we will develop an MCMC method that will asymptotically generate samples $\{(\theta^{(i)}, k^{(i)}, z_{1:k}^{(i)})\}$ from $p(\theta, k, z_{1:k}|R)$. By discarding the "nuisance" data corresponding to latent paths this procedure will result in samples $\{\theta^{(i)}\}$ distributed according to our desired target.

In order to produce samples from the joint posterior (3.1) we will consider three primary types of updates:

1. birth and death moves which mix over the trans-dimensional path space $(k, z_{1:k})$ conditioned on the policy parameters θ;
2. fixed-dimensional updates to the path $z_{1:k}$ conditioned on the parameters θ and the path length k;

3. updates to the policy parameters θ conditional on the current fixed-dimensional path $(k, z_{1:k})$.

The last two of these proposals can be performed using standard Metropolis-Hastings and are only slight modifications of approaches generally used for sampling hidden states and model parameters respectively from more standard state space models (SSMs). It is only the first proposal type that is somewhat non-standard and requires the full generality of the reversible jump algorithm introduced in the previous section. A brief overview of this algorithm can be seen in Algorithm 1.

Algorithm 1
Basic MCMC procedure for jointly sampling both trajectories and policy parameters.

1: Initialize $(\theta, k, z_{1:k})$, possibly by sampling from the prior.
2: **for** $i \geq 1$ **do**
3: *Trans-dimensional update:* propose a new k'-length trajectory $(k', z'_{1:k'})$ and accept with the proper acceptance probability.
4: *Fixed-dimensional update:* propose a new trajectory $(k, z'_{1:k})$.
5: *Parameter update:* propose new parameters θ'.
6: **end for**

3.3.1 Sampling trajectories using reversible jump MCMC

Let $(\theta, k, z_{1:k})$ denote the current state of some Markov chain targeting our desired posterior. By holding θ fixed we will now introduce a Markov transition kernel which targets the conditional posterior $p(k, z_{1:k}|\theta, R)$ and consisting of a state-dependent mixture of birth and death moves to mix over the trans-dimensional state. Let $c_{\text{birth}}(k)$ be the probability of proposing a birth move and $c_{\text{death}}(k) = 1 - c_{\text{birth}}(k)$ be the probability of proposing a death move. We can then summarize the proposal mechanism as follows:

- With probability $c_{\text{birth}}(k)$ we will perform a birth move wherein we sample a location uniformly in the interval $\{1, \ldots, k+1\}$ and propose the insertion of a new state $Z' \sim q_\theta(\cdot|z_{j-1:j})$ at position j. This results in the proposed trajectory $= (k+1, (z_{1:j-1}, z', z_{j:k}))$.

- Otherwise, with probability $c_{\text{death}}(k)$ we will perform a death move. For this move we uniformly sample a location in the interval $\{1, \ldots, k\}$ and propose killing the indexed state, resulting in the proposed trajectory $(k-1, (z_{1:j-1}, z_{j+1:k}))$.

Note, given the horizon k we can think of this as a mixture of $2k+1$ possible moves, where with probability $c_{\text{birth}}(k)\frac{1}{k+1}$ we propose a move introducing a new state at one of $k+1$ positions; with probability $c_{\text{death}}(k)\frac{1}{k}$ we propose a move killing one of k current states. In this work we will generally consider a constant probability of proposing a birth move $c_{\text{birth}}(k) = 0.5$, however in order to ensure that we don't kill off the only component in our trajectory we must enforce $c_{\text{birth}}(1) = 1$.

With these proposals defined we can now write the acceptance ratios necessary to perform reversible jump. First, we can easily see that the joint measure of paths and their proposals is defined over the space

$$(\mathcal{Z}^k \times \mathcal{Z}^{k+1}) \cup (\mathcal{Z}^{k+1} \times \mathcal{Z}^k) \tag{3.9}$$

for all $k > 0$. Note that this space is constructed such that for some measure $\lambda^k(dx, dx')$ defined over this space, if dx is a measurable subset of $\mathcal{Z}^{k+1}$ then dx' must be k-dimensional—and vice-versa. Next let λ be some measure[1] such that λ^k dominates the k-dimensional posterior and such that the proposal is dominated by λ. Due to the proposal definitions we can see that the joint measure is dominated by the image of λ^{k+1} onto the subset of (3.9) such that k of the coordinates coincide. This is due purely to the fact that we only add or delete one component. We can then define the following density with respect to this dominating measure:

$$g((k, z_{1:k}), (k', z'_{1:k'})) \\ = p(k, z_{1:k}|R, \theta) \begin{cases} c_{\text{birth}}(k)\, \frac{1}{k+1}\, q_\theta(z'_j|z_{j-1:j}) & \text{for } k' = k+1, \\ c_{\text{death}}(k)\, \frac{1}{k} & \text{for } k' = k-1. \end{cases}$$

[1] Again, generally Lebesgue or counting.

By combining this density with (3.7) we can write the acceptance probability for birth moves as $\min(1, \alpha_{\text{birth}})$ where the acceptance ratio is

$$\alpha_{\text{birth}} = \frac{c_{\text{death}}(k+1)}{c_{\text{birth}}(k)} \cdot \underbrace{\frac{p(k+1, (z_{1:j-1}, z', z_{j:k})|R, \theta)}{p(k, z_{1:k}|R, \theta)}}_{\text{posterior ratio}} \cdot \frac{1}{q_\theta(z'|z_{j-1:j})}.$$

As noted earlier, the path posteriors need only be computed up to a normalizing constant as we can see that this constant will cancel in the relevant ratio. Given the particular form of the path posterior we can write the ratio of these densities as

$$A = \begin{cases} \dfrac{\mu(z')}{\mu(z_1)} f_\theta(z_1|z') & \text{if } j = 1, \\ \dfrac{f_\theta(z'|z_k)\, r(z')}{r(z_k)} & \text{if } j = k+1, \\ \dfrac{f_\theta(z'|z_{j-1})\, f_\theta(z_j|z')}{f_\theta(z_j|z_{j-1})} & \text{otherwise.} \end{cases}$$

Similarly, the probability of accepting a death move is $\min(1, \alpha_{\text{death}})$ where we can obtain the ratio associated with this probability from the reciprocal of the birth move, i.e.

$$\alpha_{\text{death}} = \frac{c_{\text{birth}}(k-1)}{c_{\text{death}}(k)} \cdot \frac{p(k-1, (z_{1:j-1}, z_{j+1:k})|R, \theta)}{p(k, z_{1:k}|R, \theta)} \cdot \frac{q_\theta(z_j|z_{j-1}, z_{j+1})}{1}.$$

The simplifications necessary to write the posterior ratio for this acceptance probability follow directly from those in the birth step.

Reversible jump with Jacobians

Finally, in order to be complete we should also mention an alternative construction of the previous Markov chain which is based on dimension jumping moves and their associated Jacobians. This construction, used extensively by [Green, 1995, 2003], is so closely associated with the reversible jump procedure that they are often thought to be interchangeable [see e.g. Geyer, 2010]. We will see shortly that, although this construction is quite useful in

deriving reversible jump algorithms, it is unnecessary in this situation.

We will consider as above a state space that is a disjoint union $\bigcup_{k\geq 1} \mathcal{Z}^k$ where letting $\mathcal{Z} \subseteq \mathbb{R}^d$, we can see that $\mathcal{Z}^k$ is a Euclidean space of dimension kd. The equilibrium distribution of the Markov chain is then specified by an unnormalized distribution $h(x)$, which in this case is proportional to the posterior $p(k, z_{1:k}|R, \theta)$ with $x = z_{1:k}$. Next, we will consider a collection of elementary moves such that with probability $c_i(x)$ the ith such move is proposed between spaces spaces $\mathcal{Z}^{m_i}$ and $\mathcal{Z}^{n_i}$. As a result, $c_i(x)$ is only non-zero for $x \in \mathcal{Z}^{m_i} \cup \mathcal{Z}^{n_i}$.

Now, let $\mathcal{U}_i$ and $\mathcal{V}_i$ be Euclidean spaces such that $\mathcal{Z}^{m_i} \cup \mathcal{U}_i$ is of the same dimension as $\mathcal{Z}^{n_i} \cup \mathcal{V}_i$. This is the so-called "dimension matching" condition. We can now define a proposal distribution corresponding to this move such that $q_i(\cdot|x)$ is defined over $\mathcal{U}_i$ when $x \in \mathcal{Z}^{n_i}$ and defined over $\mathcal{V}_i$ for $x \in \mathcal{Z}^{m_i}$. Finally, we must specify a function g_i that maps points in $\mathcal{Z}^{m_i} \cup \mathcal{U}_i$ to points in $\mathcal{Z}^{n_i} \cup \mathcal{V}_i$ and vice-versa. We also require that g_i is its own inverse. Given the current state of the chain x, the ith proposal then proceeds by generating $u \sim q_i(\cdot|x)$ and proposing $(y, v) = g_i(x, u)$. Intuitively, u represents the random numbers that must be generated in order to move from x to y, and v the random numbers needed to move back. Meanwhile, g_i is the transformation that takes these random numbers into the actual state-space. Given this process, the acceptance ratio is given by

$$r((x, u), (y, v)) = \frac{c_i(y)\, h(y)\, q_i(v|y)}{c_i(x)\, h(x)\, q_i(u|x)} \cdot \det(\nabla g_i(x, u)). \tag{3.10}$$

Here we can see the Jacobian of the mapping g_i in the final term.

This procedure was shown by [Green, 1995] to satisfy the conditions outlined briefly in Section 3.2 for defining a proper trans-dimensional Markov chain whose invariant distribution is our desired target. We can now consider what happens were we to apply this construction to the problem of Bayesian policy search. Let the ith update be a move between $\mathcal{Z}^k$ and $\mathcal{Z}^{k+1}$ which either gives birth or kills a component at the end of the chain. Without loss of generality we will assume x is of dimension k. We can easily see that this will correspond to a birth move with a ratio of c_i terms given by

$c_{\text{death}}(k+1)/c_{\text{birth}}(k)$ and a ratio of posterior terms appears exactly as it did in the previous section. We can also see that the proposal can be written as $q(u|x) = q(u|z_k)$ where $u = z'$ corresponds to the newly proposed terminal state. As a result, due to the dimension matching condition we know that v must be zero-dimensional, and the ratio of proposals is $1/q_i(u|z_k)$, again just as in the previous section. Finally, we can see that g_i for this move is just the identity, as a result its Jacobian will be given by the identity and our full acceptance ratio appears exactly as it did above.

3.3.2 Fixed-dimensional updates

Again, let $(\theta, k, z_{1:k})$ denote the current state of the chain. We can now propose a standard (fixed dimensional) move wherein we update all or a subset of the components $z_{1:k}$ using Metropolis-Hastings or Gibbs moves. There are many design possibilities for these moves. In general, one should block some of the variables so as to improve the mixing time of the Markov chain, where we can randomly select blocks to update or systematically update all blocks at every iteration. For some block of variables indexed by a through b we can consider a Metropolis-Hastings scheme with proposals $Z'_{a:b} \sim q_\theta(\cdot|z_{a-1:b+1})$ to update this block. We can then write the acceptance probability for this update as $\min(1, \alpha_{\text{update}})$ where the associated ratio is given by

$$\alpha_{\text{update}} = \frac{p((z_{1:a-1}, z'_{a:b}, z_{b+1:k})|k, R, \theta)}{p(z_{1:k}|k, R, \theta)} \cdot \frac{q_\theta(z_{a:b}|(z_{a-1}, z'_{a:b}, z_{b+1}))}{q_\theta(z'_{a:b}|z_{a-1:b+1})}.$$

Similar to the birth/death moves of the previous subsection we can see that the posterior ratio simplifies to

$$A = \frac{f_\theta(z'_a|z_{a-1}) \prod_{n=a}^{b} f_\theta(z'_{n+1}|z'_n)}{f_\theta(z_a|z_{a-1}) \prod_{n=a}^{b} f_\theta(z_{n+1}|z_n)}$$

for $a > 1$ and $b < k$ and for symmetric proposals $q(z_{a:b}|\cdots)$. For $a = 1$ we can see that the initial transition density will be replaced with the initial state density. For $b = k$ the final transition will be replaced with the

observation or reward "density".

In designing the fixed-dimensional proposal distributions there is a wealth of possible strategies; see e.g. the recent work of [Fearnhead, 2010] for an extensive overview. The simplest approach to updating the state $z_{1:k}$ relies on *single-site updates* which only update a single component z_n. In some situations, such as discrete state and action spaces (in an analogy to approaches taken with HMMs), it can be possible to implement a Gibbs sampler which samples directly from the conditional distribution $p(z'_n|z_{n-1}, z_{n+1})$. Other approximations may also prove feasible when the full conditional cannot be sampled from, such as those based on a Taylor expansion of the log-conditional. Although easy to implement, as noted by Fearnhead (among many others) such single-site moves can often exhibit slow mixing when there is strong temporal dependence in the state process. We will come back to this point later in Section 3.4.2. Alternatively, sampling longer blocks of variables can alleviate some of these concerns, although the end-points between two blocks can often have similar difficulties. At the other end of the spectrum, it is often possible to sample the entire state sequence using its full conditional as is the case with HMMs and LDSs; again, see [Fearnhead, 2010] for further details.

Finally, conditioned on the current trajectory $(k, z_{1:k})$ we can propose a new set of policy parameters θ' from the proposal distribution $q(\cdot|\theta)$. These new policy parameters are then accepted with probability $\min(1, \alpha_{\text{param}})$ where

$$\alpha_{\text{param}} = \frac{p(\theta')}{p(\theta)} \cdot \frac{p(k, z_{1:k}|R, \theta')}{p(k, z_{1:k}|R, \theta)} \cdot \frac{q(\theta|\theta')}{q(\theta'|\theta)}.$$

3.3.3 Preliminary experiments

In this section we now present preliminary experiments illustrating the use of MCMC algorithms for Bayesian policy search. It should be noted from the outset, however, that these results are mainly for illustrative purposes and will show both some of the strengths and weaknesses of the proposed approach. In the next section we will present additional modifications to ad-

dress some of these weaknesses, as well as more comprehensive experiments.

We first consider a linear-Gaussian MDP model, similar to that in Section 2.3, namely

- state and action spaces $\mathcal{X} = \mathcal{A} = \mathbb{R}^2$;
- zero-mean, Gaussian initial states $\mu(x_1) = \mathcal{N}(x_1; 0, \sigma_1^2 I)$;
- linear-Gaussian transitions $f(x_{n+1}|x_n, a_n) = \mathcal{N}(x_{n+1}; x_n + a_n, \sigma_2^2 I)$;
- squared-exponential rewards $r(x) = \exp(-0.5 \|x - m\|^2 / \sigma_3^2)$.

In a departure from the previous section, however, we define the following stochastic policy

$$A_n = (w + \delta_n) \begin{bmatrix} \cos(\theta + \omega_n) \\ \sin(\theta + \omega_n) \end{bmatrix}.$$

Here we have defined a policy controlled only via the parameters $\theta \in [0, 2\pi]$; w is a small constant step-size; δ_n and ω_n are small uncontrolled, normally distributed perturbations. Intuitively, this policy corresponds to choosing a direction θ in which the agent will walk. While unrealistic from a real-world perspective, this allows us a method to easily evaluate and plot the convergence of our algorithm. An illustration of this space can be seen in Figure 3.1 where $m = [1, 1]^T$.

The primary choices affecting the performance of this algorithm concern the proposal distributions for the birth moves, the fixed-dimensional updates, and finally the parameter updates. To a lesser degree the prior also affects the algorithm performance, however for this model we made the straightforward choice of an uninformative, uniform prior over $[0, 2\pi]$. For birth moves with $1 < j < k$ we propose a new state normally distributed equidistant between the two neighboring points; for $j = 1$ or $j = k$ we sample directly from the path prior. We used single-site moves to update the fixed-dimensional path, where the proposals were "smoothed" towards the current path θ, plus some additional Gaussian noise. Finally, the parameter updates were simple random walk Metropolis-Hastings with a Gaussian kernel.

Not only can we use this to sample from the desired posterior, and thus sample proportional to the expected reward, but in some situations

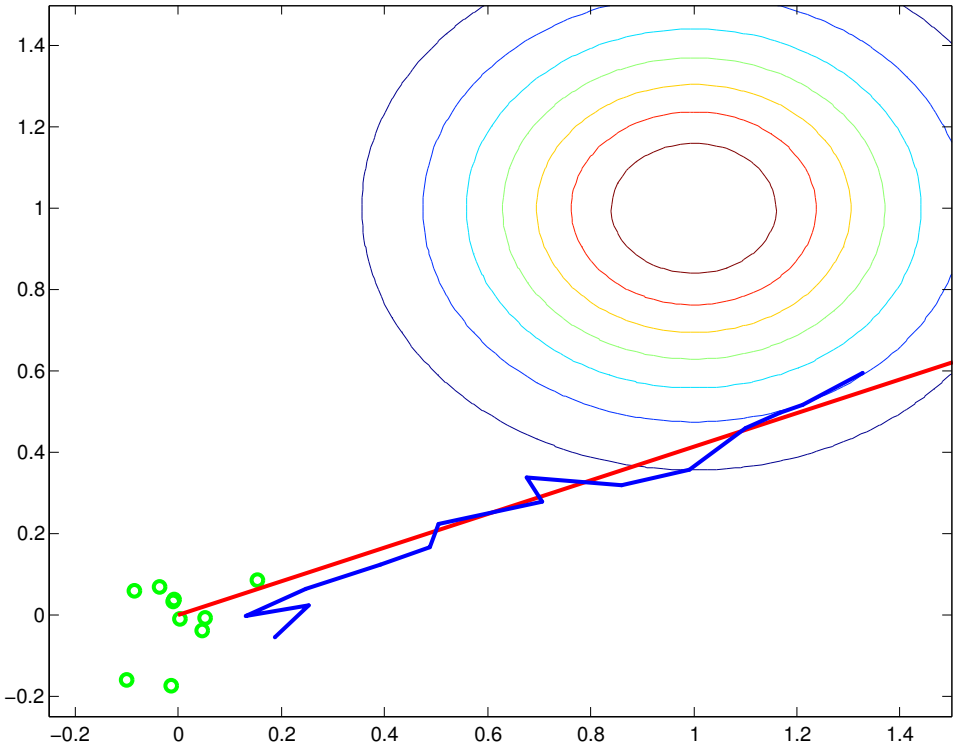

Figure 3.1: Illustration of the 2-dimensional angular-policy space used to demonstrate of reversible jump policy search. Ten sample points are shown distributed according to μ and the contour plot corresponds to the reward function r. The red line denotes the policy parameterized by some angle θ, while a path is drawn in blue sampled from this policy.

we can use it to identify the optimal parameters. We can easily see that the distribution in question is unimodal and symmetric, and as a result the mean and the mode coincide. In Figure 3.2 we show the convergence of this approach. We also compare against the PEGASUS policy search method of [Ng and Jordan, 2000]. Here we see that the Bayesian method is able to mix over the space more quickly and quickly settles on the optimal policy parameters. We found that this was largely due to its ability to move more quickly over regions where the gradient is close to zero.

It is crucial to note however that the MCMC method required substantially more tuning: we found that the birth and death moves were tricky to obtain proper mixing due to the high correlations between subsequent states. We should also point out that the particular fixed-dimensional path updates used in this section required significant prior knowledge of the policy's behavior. This last point is fairly important in that we can look at the MCMC method developed in this section as less of a "black-box" MDP solver and more of a recipe for methods which identify regions of high reward, albeit a recipe that requires more user knowledge in order to achieve better performance. In the next section we will introduce a number of extensions to the basic reversible jump procedure in order to address these concerns.

3.4 Improved inference strategies for reversible jump policy search

In the previous section we introduced an MCMC procedure to sample the components of a parameterized policy with probability proportional to the expected rewards of that policy. The behavior of this sampler was, however, highly dependent on the choice of proposal distribution, with little advice given on how to select this proposal. In this section we will introduce a number of modifications to both the sampler itself and the likelihood model associated with the MDP in order to improve the behavior of the sampling procedure.

3.4.1 Utilizing the entire reward sequence

In Section 2.1 we introduced an observation model $p(R|k, z_k)$ such that the probability of our "dummy observation" $R = 1$ is given by the reward at the end of the latent chain, i.e. $r(z_k)$. This formulation was crucial to the development of the efficient EM-based procedures in this chapter, as well as the earlier work of as [Toussaint and Storkey, 2006]. In the case of sampling-based procedures, however, we can greatly improve upon earlier methods by instead using the entire sequence of rewards $\{r(z_n)\}_{n \leq k}$.

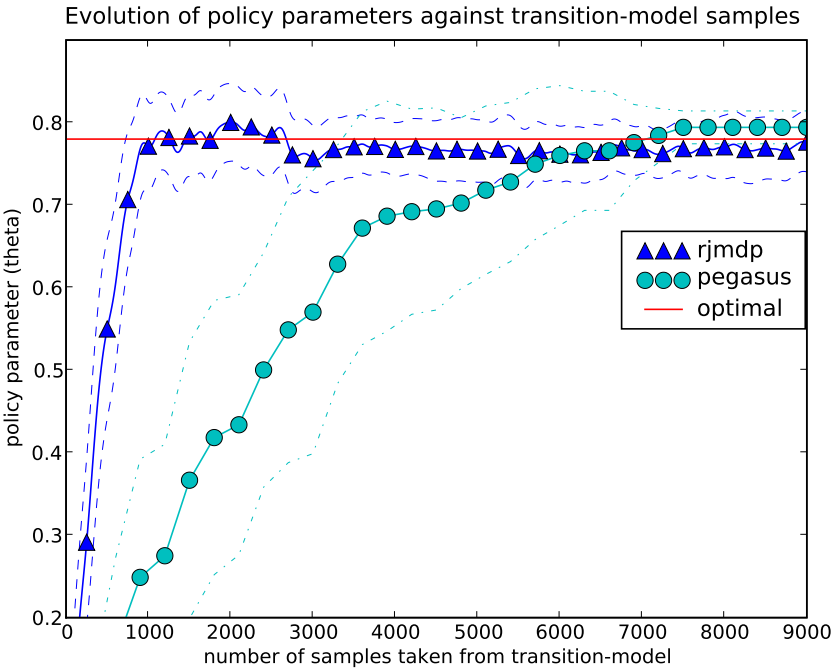

Figure 3.2: Convergence of reversible jump policy search compared against PEGASUS for the preliminary angular-policy problem. and our Bayesian policy search algorithm Both algorithms start from $\theta = 0$ and converging to the optimum of $\theta^* = \pi/4$. The plots are averaged over 10 runs. For the reversible jump algorithm we plot samples taken directly from the MCMC algorithm itself: plotting the empirical average would produce an estimate whose convergence is almost immediate, but we also wanted to show the "burn-in" period. For both algorithms lines denoting one standard deviation are shown and performance is plotted against the number of samples taken from the transition model.

In fact, the use of rewards only at the end of the chain can prove harmful in terms of the mixing properties of the underlying Markov chain. Consider, for example an MDP with two regions of high immediate reward but areas of relatively low reward between them. We can then consider what happens when the current state of the Markov chain has z_k landing in one of these regions and a birth proposal is made after step k with z^* leaving this region. The acceptance probability will be given by $\alpha_{\text{birth}} \propto \gamma r(z^*)/r(z_k) \approx 0$, where we have ignored those terms not contained in the posterior ratio. In this situation, the regions of high reward will be much higher than the surrounding low reward regions and as a result the probability of accepting such a birth move will be quite low. Often the probabilitiy of accepting birth moves (and respectively death moves) will become unacceptably low and lead to very slow mixing of samples from the state space, under which it may prove very difficult to distinguish between "good" and "bad" policies. An illustration of this is shown in Figure 3.3.

Intuitively, Figure 3.3 suggests that the reason behind this poor mixing is that we are not using enough of the reward information when deciding whether to accept or reject proposals. In order to alleviate this problem, we will instead introduce an alternative likelihood model that depends on the entire path, namely one that is given by the sum of undiscounted rewards,

$$p(R = 1|k, z_{1:k}) = \sum_{n=1}^{k} r(z_n). \tag{3.11}$$

Again, just as in Section 2.1, the actual value of the dummy observation $R = 1$ is arbitrary, and as a result we will continue to drop the value of this "observation". We also emphasize the functional form of r is exactly the reward model given in the MDP specification. We are only introducing a dummy observation R with likelihood given by (3.11) in order to *solve* this problem using MCMC. Now, given the newly introduced likelihood we can make the following claim:

Proposition 4. *Given the observation model defined in* (3.11) *the marginal likelihood* $p(R = 1|\theta)$ *of the policy parameters under these observations is*

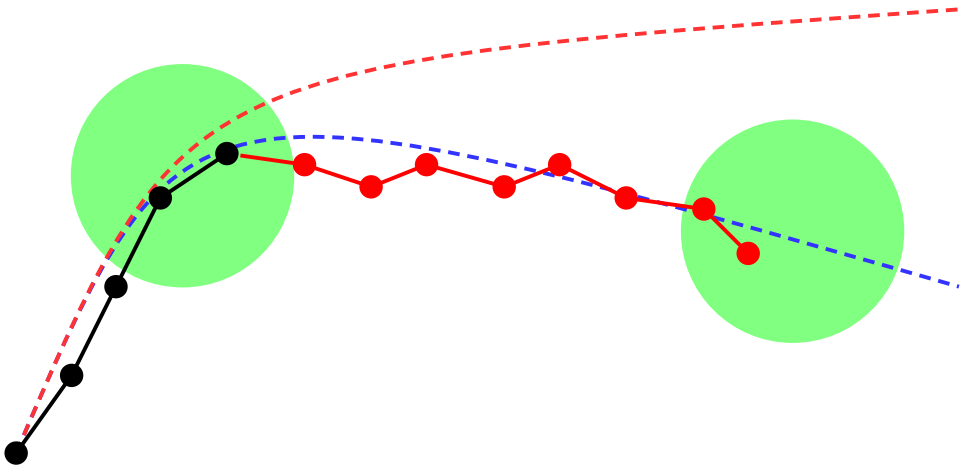

Figure 3.3: Illustration showing the poor mixing properties that can occur when only using rewards at the end of a chain. Let the green circular areas denote regions of high reward with the black dots denoting the current state of the chain. Consider then proposed states given by the red dots. If only one such dot is proposed, this proposal will have very low probability of being accepted. If the two dashed lines represent two alternative policies then it becomes difficult to distinguish these policies (and hence sample them) based only on the current state. Note that one of these policies is "good" in that it passes through the second high reward area whereas the other policy is "poor" and hence should be sampled less.

equivalent to the expected reward $J(\theta)$.

Proof. Just as we did when proving Proposition 1 we can start from the expected reward and rearrange the terms that depend on γ,

$$J(\theta) = (1-\gamma)\sum_{n=1}^{\infty}\frac{\gamma^{n-1}}{1-\gamma}\int r(z_n)\,p(z_{1:n}|\theta)\,dz_{1:n}.$$

By using the series expansion of $\gamma^n/(1-\gamma)$ we can write this as

$$= (1-\gamma)\sum_{n=1}^{\infty}\Big(\sum_{k=n}^{\infty}\gamma^{k-1}\Big)\int r(z_n)\,p(z_{1:n}|n,\theta)\,dz_{1:n}$$

and finally, by changing the order of summation, we obtain the marginal posterior as desired

$$
\begin{aligned}
&= (1-\gamma)\sum_{k=1}^{\infty}\gamma^{k-1}\sum_{n=1}^{k}\int r(z_n)\,p(z_{1:n}|\theta)\,dz_{1:n} \\
&= \sum_{k=1}^{\infty}\int\left(\sum_{n=1}^{k} r(z_n)\right)p(k, z_{1:k}|\theta)\,dz_{1:k}. \qquad \square
\end{aligned}
$$

It is interesting to note that equality now holds for this formulation, rather than being proportional as in the previous formulation. Further, we must emphasize that although the discount factor still appears in the time prior $p(k)$, the sum of rewards used in formulating this new likelihood model is *undiscounted*.

This reformulation allows us to introduce the new target distribution

$$
p(k, z_{1:k}, \theta|R) \propto p(R|k, z_{1:k})\,p(k, z_{1:k}|\theta)\,p(\theta), \tag{3.12}
$$

which we will use throughout the rest of this chapter. Just as in the previous section we can easily see that this obtains the desired marginal target, due to Proposition 4. Finally, we can return to the thought experiment introduced above wherein the current state of the Markov chain lies in a region of high reward and a proposal is made which leaves this region. In this case the resulting acceptance probability will be

$$
\alpha_{\text{birth}} \propto \gamma\frac{r(z_1)+\cdots+r(z_k)+r(z^*)}{r(z_1)+\cdots+r(z_k)} \approx \gamma.
$$

In other words, the probability of accepting a new state outside of the region of high reward will be controlled primarily by the discount factor. This will in turn lead to better mixing, a property that we will see in later experiments.

We should also note that this same approach can be used when sampling in general. For example, [Vlassis and Toussaint, 2009] extend the EM procedure of the previous chapter to the Monte Carlo setting, and here they in parallel develop a similar approach to the problem.

3.4.2 Explicit noise variables

While theoretically sound, sampling from (3.1) or (3.12) often requires a carefully tuned proposal distribution in order to accurately explore the posterior. In many cases the policy parameters θ and the sequence of state/action pairs $z_{1:k}$ (as well as the individual steps within that sequence) will be highly correlated, resulting in a Markov chain which mixes very poorly over these variables. Blocking the variables can improve the mixing time of the Markov chain, but this does not completely alleviate the problem as the blocks themselves can exhibit high correlations, especially at the end-points of each block. Here, however, we adopt an even more efficient sampling strategy.

In many models both the transition model and the policy take the form of deterministic functions of the current state and action as well as some additional vector noise variables. We will write these models as

$$\begin{aligned} A_n &= \pi_\theta(x_n, \phi_n), \\ X_n &= f(x_{n-1}, a_{n-1}, \psi_n) \end{aligned}$$

i.e. where $\epsilon_n = (\phi_n, \psi_n)$ denotes the noise (i.e. stochastic) components distributed according to

$$p(\epsilon_n|\theta) = p(\psi_n)\, p(\phi_n|\theta).$$

Only by not conditioning on the noise variables ϵ_n do we obtain the standard stochastic transition models; conditioned on these variables the system is entirely deterministic. Finally, under these assumptions the initial-state is a special case, however it becomes notationally convenient to consider this as "fully stochastic" and write $\psi_1 = x_1$. Here we allow the noise ϕ_n to depend upon θ so that the policy can control exploratory noise. In more general settings it might also make sense to let x_1 depend upon θ, but here we assume that the initial state is uncontrolled. An illustration of this model can be seen in Figure 3.4.

Under these circumstances, the strong correlation exhibited by z_n and

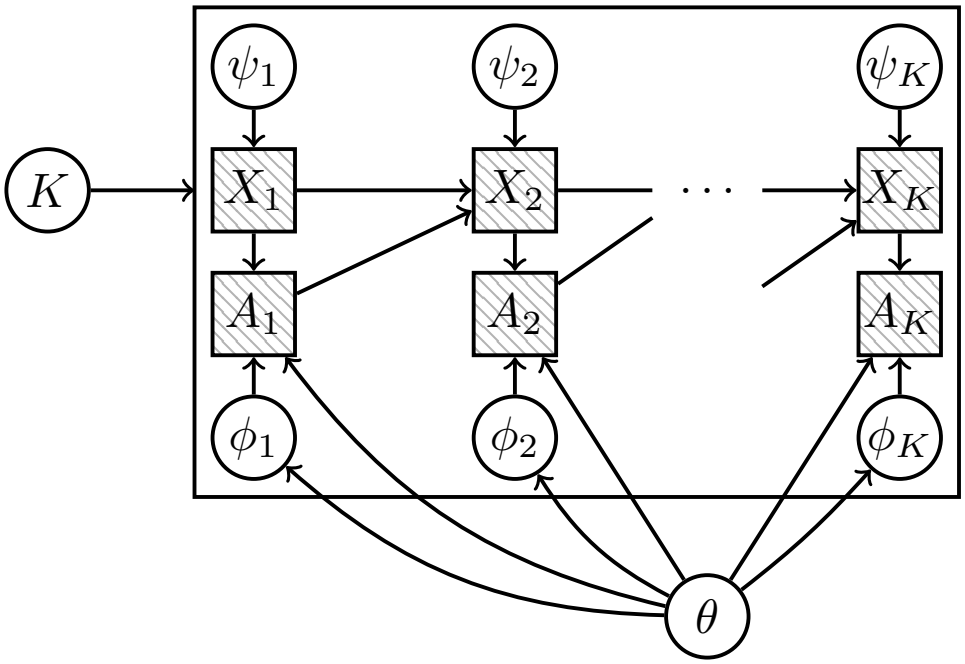

Figure 3.4: Graphical model depicting the use of auxiliary noise variables. Here the square, shaded nodes are deterministic given their parents.

z_{n+1} is mostly due to the deterministic components. We remind the reader that it is this strong correlation that causes any MCMC algorithm to mix poorly. We can limit this problem by sampling $\epsilon_{1:k}$ rather than the state/path terms. This is an idea closely related to the techniques discussed by [Papaspiliopoulos et al., 2003]. Under this re-parameterization, the new target distribution is

$$p(k, \epsilon_{1:k}, \theta|R) \propto p(R|k, z_{1:k})\, p(k, \epsilon_{1:k}|k, \theta). \tag{3.13}$$

Although we have eliminated the need to sample z_n, we must still calculate it in order to compute the reward at time n; *this calculation is, however, deterministic given* z_{n-1} *and* ϵ_n.

The reformulation mitigates the mixing problems due to the strong dependencies between θ and $z_{1:k}$ as well as between z_{n+1} and z_n that were

caused by the deterministic components of the policy and transition densities. The dependencies between the variables in this new artificial joint distribution are purely due to the reward function r and in many cases will be comparatively weak.

Apart from its decorrelating effect, this technique has a secondary benefit as a variance reduction technique. The noise terms $\epsilon_{1:k}$ can act as *common random numbers*, in a way that is closely related to the idea of fixing random seeds in policy search [Ng and Jordan, 2000]. In particular, we can fix the noise variables for a predetermined number of MCMC moves updating the policy. In doing this, both $\theta^{(i)}$ and $\theta^{(i-1)}$ will depend on the same random seeds (noise terms). Consequently, the variance of the policy update will be reduced. This is a direct consequence of the fact that the variance of the difference of two estimators based on Monte Carlo simulations is equal to the sum of the individual variances of each estimator minus their joint covariance [see Spall, 2005].

3.5 Marginal Optimization

So far we have provided a method for sampling from a posterior distribution, either (3.1) or (3.12), that exhibits a marginal over policy parameters proportional to the expected reward under those parameters. Our goal, however, is often to estimate the maximum $\theta^* = \arg\max_\theta J(\theta)$. As noted earlier, if $J(\theta)$ happens to have a strongly dominant and highly peaked mode around the global maximum θ^*, we can justify sampling from $p(k, z_{0:k}, \theta|R)$ and deriving a point estimate of θ^* by averaging the resulting sampled parameters. However, in general the assumption of such a favorable $J(\theta)$ is unrealistic. If $J(\theta)$ is multimodal or fairly flat then this approach will yield poor estimates.

Instead, in this section we will present two modifications to address this problem. The first uses annealing to concentrate the samples directly on the modes. This step is, however, complicated by existence of the latent data: we have to ensure that we properly anneal our distribution in the marginal setting. We will then introduce "clean-up" step involving clus-

tering that helps in the event of multiple modes with similar performance characteristics.

3.5.1 Annealing

We will now consider the following marginal density

$$p_\nu(\theta) \propto J(\theta)^\nu \, p(\theta).$$

For large exponents ν the probability mass of this distribution will concentrate on the global maximum θ^*. If we could sample from $p_\nu(\theta)$, then the generated samples would allow us to derive a much better point estimate of θ^*. Note however that this is not as simple as it might seem at first glance. For example raising the joint density in (3.12) to the power of ν will not result in a distribution with this desired marginal.

A method for generating samples from marginal distributions of this form was proposed by Müller et al. [1998, 2004] in the context of optimal design and independently in [Doucet et al., 2002] in the context of marginal maximum a posteriori estimation. The trick is to define an artificial distribution jointly over multiple simulated trajectories. To simplify notation let us first define $\zeta_j = (k_j, z_{0:k_j})$ to represent one simulated trajectory. For integer values of ν we can then define the following artificial target distribution

$$p_\nu(\theta, \zeta_{1:\nu}|R) \propto p(\theta) \prod_{j=1}^{\nu} p(R|\zeta_j) \, p(\zeta_j|\theta). \tag{3.14}$$

By jointly marginalizing over the ζ_j for each of the $j \leq \nu$ different trajectory instantiations we can easily to verify that this distribution does indeed admit the desired marginal distribution $p_\nu(\theta) \propto J(\theta)^\nu \, p(\theta)$. However, because the modes of $J(\theta)^\nu$ will typically be narrow and widely separated for large ν, sampling from this distribution using Markov chain Monte Carlo techniques directly is difficult.

We therefore, we take a simulated annealing approach [as in Doucet et al., 2002, Müller, 1998, Müller et al., 2004] in which we start sampling

from p_ν with $\nu = 1$, and then slowly increase ν over time according to an annealing schedule. Increasing ν slowly enough allows the chain to efficiently explore the whole parameter space before becoming more constrained to the major modes for larger values of ν.

One limitation of Equation (3.14) is that the annealing schedule is limited to full integer steps. However, we can further generalize this approach to allow for a real valued annealing schedule by defining the modified target distribution

$$p_\nu(\theta, \zeta_{1:\lceil\nu\rceil}) \propto p(\theta) \left(\prod_{j=1}^{\lfloor\nu\rfloor} p(R|\zeta_j)\, p(\zeta_j|\theta) \right) p(R|\zeta_{\lceil\nu\rceil})^{\nu - \lfloor\nu\rfloor} p(\zeta_{\lceil\nu\rceil}|\theta), \qquad (3.15)$$

where ν is now real valued and $\lfloor\nu\rfloor$ and $\lceil\nu\rceil$ denote the integer valued floor and ceiling of ν. For integer values of ν, this distribution again admits the marginal $p_\nu(\theta) \propto J(\theta)^\nu\, p(\theta)$ as before. While this does not hold for the intermediate distributions with real valued ν, these distributions provide a smooth bridge between the integer steps. This allows for more gradual annealing, thereby reducing the variance of the overall sampler.

While in theory we should let ν approach infinity, in practice this is not computationally feasible. Instead we can choose an annealing schedule that plateaus at a final integer value $\nu_{\max}$, at which point the chain is run for another M iterations. The last M samples, marginally distributed from $p_{\nu_{\max}}(\theta)$, can then used as the basis of a point estimate of θ^*.

3.5.2 Clustering

If $J(\theta)$ has a unique maximum and $\nu_{\max}$ is sufficiently large, the final samples from $p_{\nu_{max}}(\theta)$ will be concentrated around θ^*. In this case averaging the L final samples can provide a good estimate of θ^*. In practice however, it is possible that $p_{\nu_{max}}$ still has multiple modes with significant probability mass. In this case simple averaging can lead to a poor estimate.

To provide a better point estimate under these circumstances we can cluster the final samples and use the center of the largest cluster as our estimate of θ^*. For the clustering we use simple agglomerative clustering

using average linkage (UPGMA). Other techniques such as for example mean shift clustering [Cheng, 1995] could be used instead. Note however that the popular K-Means algorithm is not suited for this purpose as it tends to split high density areas into multiple clusters.

3.6 Experiments

3.6.1 Linear-Gaussian models

We first experiment with linear-Gaussian transition models of the same form as those in Section 2.3, i.e.

$$f(x_n, u_n) = Ax_n + Bu_n + \mathcal{N}(0, \Sigma), \text{ and}$$
$$\pi_\theta(x_n) = Kx_n + m \text{ for } \theta = (K, m).$$

This model is particularly interesting if we allow for multimodal rewards, as this will in general induce a multimodal expected reward surface. Figure 3.5 contrasts samples taken from both the non-annealed and annealed distribution (with annealing factor $\nu = 20$) on a model with 1D state- and action-spaces. In this example we can see that the simple approach of averaging samples $\{\theta^{(i)}\}$ results in a very poor estimate of the policy parameters, whereas both clustering and annealing are correctly able to recover the optimum.

3.6.2 Particles with force-fields

For a more challenging control problem we chose to simulate a physical system in which a number of repellers are affecting the fall of particles released from within a start region. The goal is to direct the path of the particles through high reward regions of the state space in order to maximize the accumulated discounted reward. The four-dimensional state-space in this problem consists of a particle's position and velocity $(p, \dot{p})$ for $p \in \mathbb{R}^2$. Actions consist of repelling forces acting on the particle. Additionally, the particle is affected by gravity and a frictional force resisting movement.

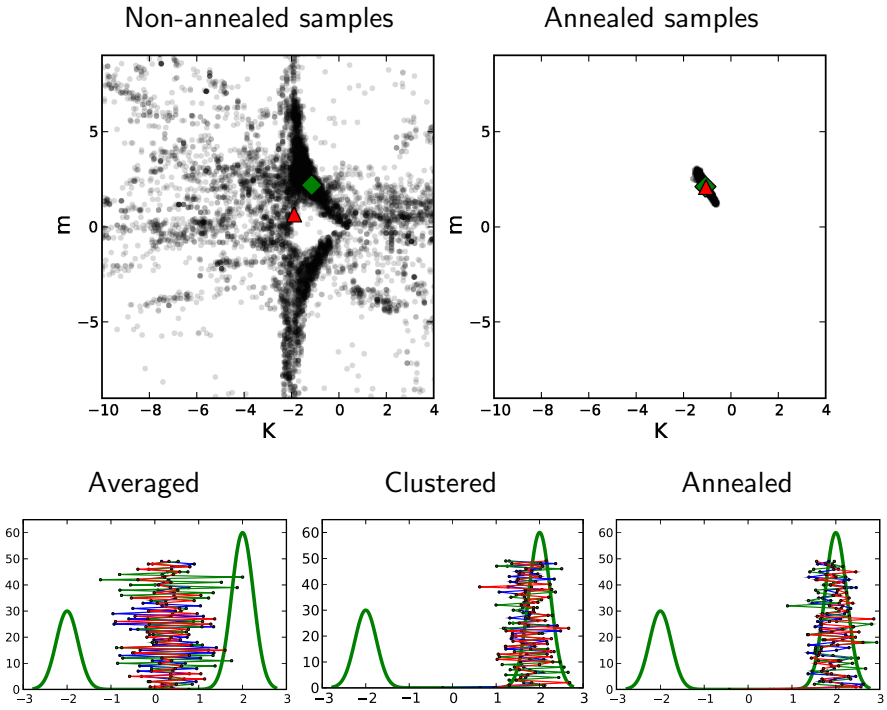

Figure 3.5: Policy parameters sampled from a 2-dimensional linear-Gaussian MDP with multimodal rewards. Shown in the top row are samples both with annealing (right) and without (left). Simple averaging of the sampled parameters leads to the estimates given by the red triangle, whereas the green diamonds are the point estimate found by clustering these same samples. Also shown in the second row are sample trajectories under these different estimates where the y-axis gives the discrete time index.

The deterministic policy is parameterized by k repeller positions r_i and strengths w_i with a functional form given by

$$\pi_\theta(p, \dot{p}) = \sum_{i=1}^{k} w_i \frac{p - r_i}{\|p - r_i\|^3}.$$

That is, each repeller pushes the particles directly away from it with a force inversely proportional to its distance from the particle. In our experiments

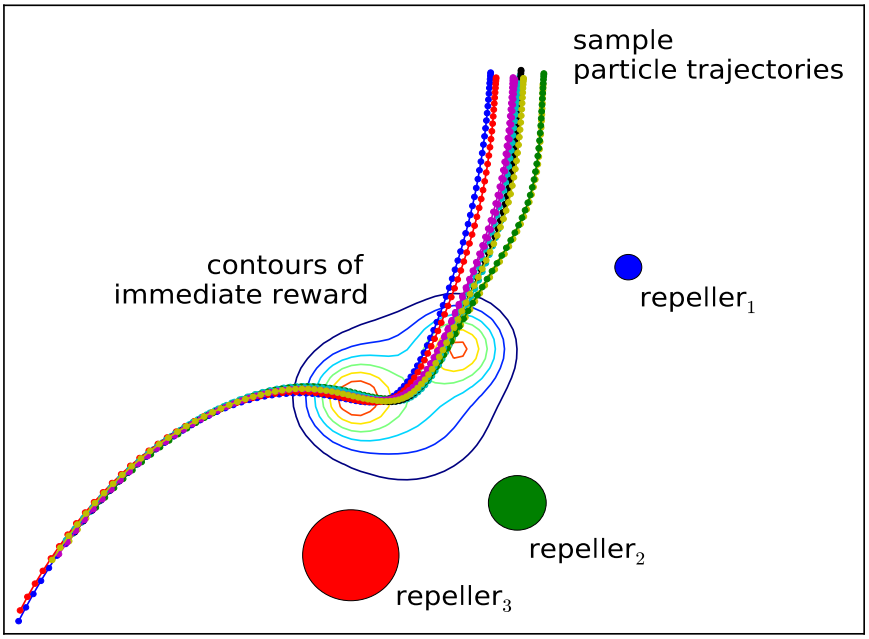

Figure 3.6: Example of the particle control problem with force-fields. Shown are contours of the immediate reward, a an example policy—not necessarily optimal—consisting of 3 repellers, and a number of trajectories sampled from this model.

the particle's start position is uniformly distributed within a rectangular region. At each time step the particle's position and velocity are updated using simple deterministic physical forward simulation and a small amount of Gaussian transition noise is added to the particle's velocity. See Figure 3.6 for an example of this model.

In Figure 3.7 we use this particle model to show the benefits of the summed likelihood formulation (3.12) over the target distribution which only uses rewards at the last time step (3.1). We employ the noise variable parameterization and the annealing and clustering techniques discussed in Sections 3.4.2 and 3.5 in both samplers. The reward model used in this example is composed of multiple circular reward zones. A high constant reward is awarded inside these zones and close to no reward outside. Note

that the discontinuous and multimodal nature of this reward surface makes for a very challenging control problem. In this and the following experiments we are searching for the optimal placement and strengths of two repellers, resulting in a 6 dimensional control problem. In our implementation we are updating the 6 policy components in 4 blocks for the positions and strengths of the two repellers.

When evaluating the reward at the last step only, the sampler has difficulties crossing the gaps between the reward zones, as indicated by the relatively low acceptance ratios of birth and death moves, see the second plot of Figure 3.7. This leads to the sampler getting stuck in local minima, resulting in poor policy estimates. The summed rewards formulation on the other hand allows for better mixing over path lengths, making it more likely to find the high reward zone at the bottom. This ultimately results in much better policies. Note how the policy found using our summed rewards approach, visualized in the bottom-right of this figure, uses the two repellers to not only direct the particles towards the high reward zone but to also slow them down inside this zone in order to accumulate as much reward as possible.

Figure 3.8 compares the algorithm described in this paper with the PEGASUS technique of [Ng and Jordan, 2000] using numerically computed gradients. In particular we are interested in learning using deterministic policies, and PEGASUS can be used directly in this setting. We compared 10 runs of each algorithm on the particle system model shown in the bottom two subplots, where the reward model is a single Gaussian in position-space. Even though the reward model is unimodal, the resulting expected reward surface is highly multimodal: two such modes are displayed in the bottom two subplots. The poor performance and high variance of PEGASUS is mainly due to these local maxima, as well as plateaus in the reward surface.

Finally, Figure 3.9 uses the same problem from the previous experiment to compare the sampler based on the noise-variable formulation with the basic reversible jump approach introduced earlier. By examining the resulting policy estimates we can see that the proposed reformulation significantly outperforms the previous method on this model. This results from the older

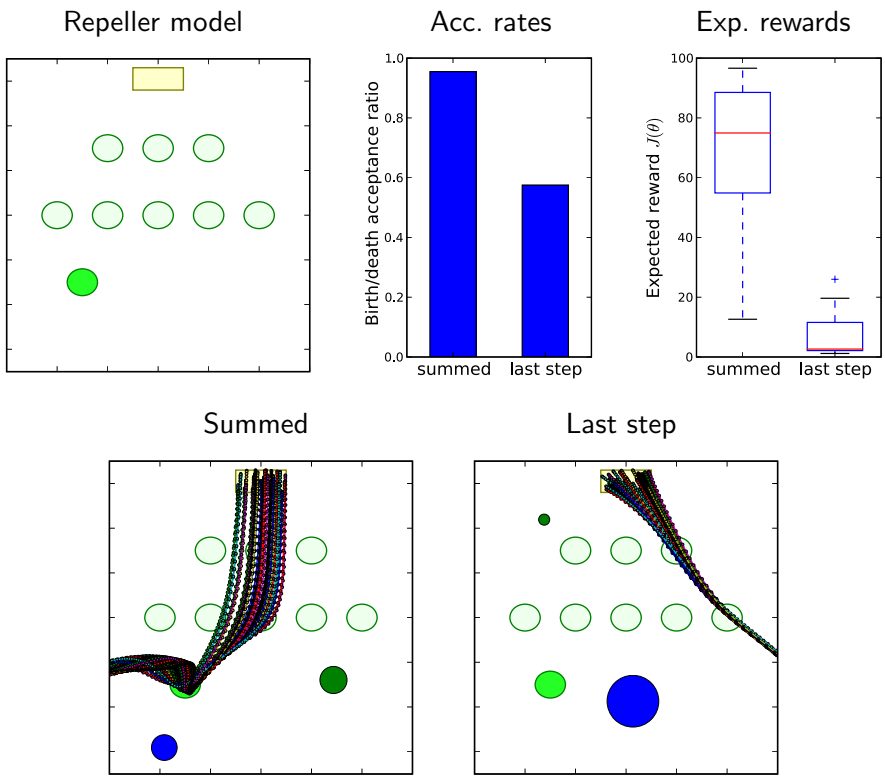

Figure 3.7: Comparison of Bayesian policy search using the summed and last-step likelihoods. The problem, shown in the top left, features multiple reward zones, with the bottom-most zone yielding a 50 times higher reward than the others. We also display the average acceptance ratios and compare the expected rewards for the policies found using 10 runs of each sampler. The final two plots visualize two of the computed policies; one for the summed reward formulation (bottom-left) and one when only evaluating the reward at the last step (bottom-right).

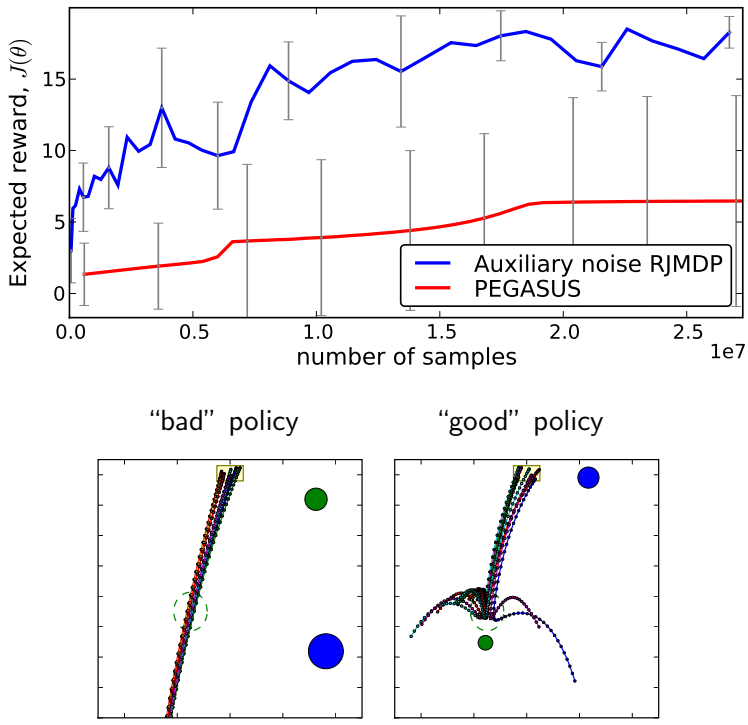

Figure 3.8: Comparison of Bayesian policy search with PEGASUS on the repellers model averaged over 10 runs, where error-bars display one standard deviation. The x-axis displays the number of samples taken from the transition model. Also shown are a "bad" local maxima found by PEGASUS, and a "good" policy found by our sampler.

method's poor mixing over trajectories, as evidenced by the extremely low acceptance rate for path updates. In order to explore the space of trajectories at all, this method therefore needs to shrink trajectories using death moves and subsequently re-grow them using birth moves. However, the acceptance ratios for such birth and death moves are themselves significantly lower than for our proposed sampler, rendering this way of mixing in trajectory space inefficient as well.

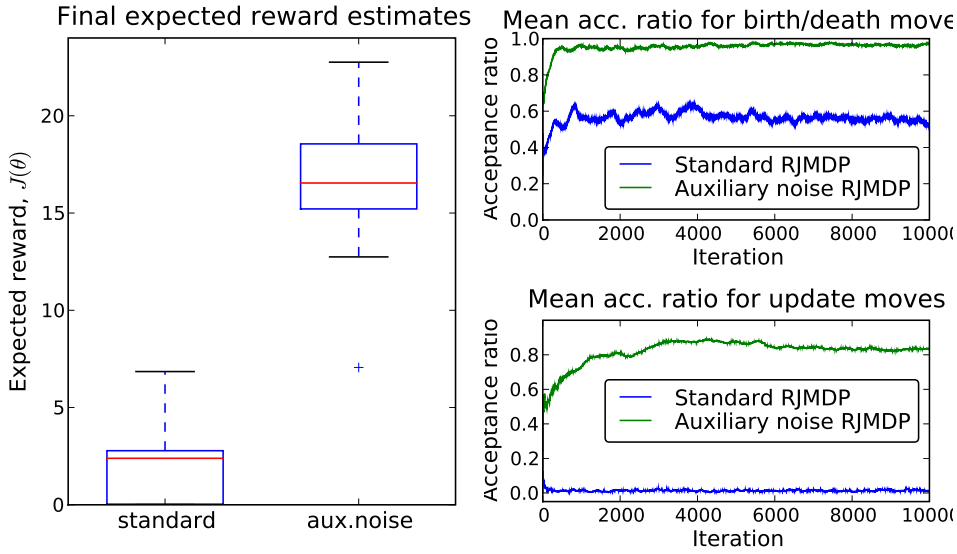

Figure 3.9: Comparison between the explicit noise variable approach and the basic MCMC approach on the particle system model. The left-most plot shows the expected rewards for the final policies found by both methods across 10 runs. The right plots display the averaged acceptance ratios for birth and death moves and the acceptance ratios for trajectory update moves.

3.7 Chapter summary and conclusions

In this chapter we presented what was at the time the first application of reversible jump MCMC, and MCMC in general, to the problem of policy search. We then provide preliminary results showing that this method can be effective in practice. However, the proposal mechanism necessary to obtain these results required careful hand-tuning which is not practical in many problems of interest.

We then presented several important improvements to this approach, specifically targeting the problem of poor mixing of the Markov chain. The experiments provide clear evidence that the proposed modifications are needed to attack higher-dimensional stochastic decision problems. In particular, the experimental results show that significant improvements are obtained when incorporating more reward information into the likelihood

model (Figure 3.7) and when using the explicit noise variables to break state-space dependencies and reduce variance (Figure 3.9). It is also clear that the proposed simulated annealing and clustering techniques allow us to find better point estimates of the optimal policy (Figure 3.5). Finally, we observed favorable performance of the proposed approach in comparison to state-of-the-art techniques such as PEGASUS (Figure 3.8). As long as there is structure in the state space, one can adopt standard Rao-Blackwellization and blocking techniques to efficiently carry out inference in the Bayesian network. The main difficulty here lies in dealing with the dimensionality of the policy space, where often there seems to be much less structure to exploit.

Finally, this chapter shows that this method compares favorably to gradient-based methods for models with with many modes and plateaus. However, in order to ensure the proper mixing of the Markov chain we must be willing to accept a sizeable number of rejection steps, which may prove inefficient as an optimization mechanism. As a result, in Chapter 6 we will present another gradient-free mechanism that is more sample efficient. It may prove that the benefits of the algorithms presented in this chapter are more beneficial due to their Bayesian qualities, e.g. as a way to incorporate prior information as well as provide a full "posterior" over the policy parameters. This can be useful, for example, in situations where the models are estimated from data as well, and their is also underlying uncertainty in these measurements.

Chapter 4

Regularized Least Squares Temporal Difference learning with nested ℓ_2 and ℓ_1 penalization

In previous chapters we have focused on the problem of policy search by directly parameterizing the control policy of an MDP and optimizing these parameters in order to maximize the expected reward under that policy. In this chapter we will consider an *indirect*—although perhaps more classical—approach to parameterizing the policy that is instead in terms of the value function. In other words we will instead consider the task of estimating a function which returns the value of being in some state $x \in \mathcal{X}$. In this chapter, we also attack the problem of reinforcement learning (RL), wherein rather than assuming a model of the control task, we can instead only sample states, actions, and rewards from the underlying MDP.

In the setting of RL, least-squares temporal difference learning (LSTD) as introduced by [Bradtke and Barto, 1996] is a very popular mechanism for approximating the value function V^π of a given policy π. More precisely, this approximation is accomplished using a linear function space $\mathcal{F}$ spanned by

a set of k features $\{\phi_i\}_{i=1}^k$. Here, the choice of feature set greatly determines the accuracy of the value function estimate. In practice, however, there may be no good, *a priori* method of selecting these features. One solution to this problem is to use high-dimensional feature spaces in the hopes that a good set of features lies somewhere in this basis. This introduces other problems, though, as the number of features can outnumber the number of samples $n \leq k$, leading to overfitting and poor prediction. In the linear regression setting, regularization is commonly used to overcome this problem. The two most common regularization approaches involve using ℓ_1 or ℓ_2 penalized least-squares, known as *Lasso* or *ridge regression* respectively [see e.g. Friedman et al., 2001]. The approach of Lasso is of particular interest due to its *feature selection* property, wherein the geometric form of the ℓ_1 penalty tends to encourage sparse solutions. This property is especially beneficial in high-dimensional problems, where we have many more features than samples, as they allow solutions to be expressed as a linear combination of a small number of features.

The application of the Lasso to the problem of value function approximation in high dimensions would thus seem to be a perfect match. However, the RL setting differs greatly from regression in that the objective is not to recover a target function given its noisy observations, but is instead to approximate the fixed-point of the Bellman operator given sample trajectories. The addition of this fixed-point creates difficulties when attempting to extend Lasso results to the RL setting. Despite these difficulties, one particular form of ℓ_1 penalization has been previously studied, both empirically [e.g., Johns et al., 2010, Kolter and Ng, 2009] and theoretically in [Ghavamzadeh et al., 2011], with interesting results.

In this chapter, we consider the two-level nested formulation of LSTD used by [Antos et al., 2008] and by [Farahmand et al., 2009]. Under this formulation, the first level optimization is defined by the projection of a Bellman image onto the linear space $\mathcal{F}$, and the second level accounts for the fixed point part of the algorithm. This formulation allows us to define a wide range of algorithms depending on the specific implementation of the projection operator and the fixed point step. In particular, we will discuss

a number of regularization methods for LSTD, two based on ℓ_2 penalties (one of which has been introduced by Farahmand et al.), one based on an ℓ_1 penalized projection (as introduced by Kolter and Ng), and finally we will introduce a novel approach which solves the problem via two nested optimization problems including an ℓ_2 and an ℓ_1 penalty (L_{21}). Unlike previous methods using ℓ_1 regularization in LSTD, this new approach introduces the ℓ_1 penalty in the fixed point step and this allows us to cast the problem as a standard Lasso problem. As a result, we do not need to develop any specific method to solve the corresponding optimization problem—as needed by Kolter and Ng, where a specific implementation of LARS has been defined—and we can apply general-purpose solvers to compute its solution. This additional flexibility also allows us to perform an explicit standardization step on the data similar to what is done in regression. We show in the experiments that all the methods using ℓ_1 regularization successfully take advantage of the sparsity of V^π and avoid overfitting. Furthermore, we show that, similar to regression, standardization can improve prediction accuracy, thus allowing L_{21} to achieve a better performance than LARSTD of Kolter and Ng.

Finally, we should note that in this chapter we will focus on the problem of computing the value function as a method of policy evaluation, and consider regularization schemes within this paradigm. In order to apply these methods to reinforcement learning tasks we would need to extend this approach to learning state-action value functions in order to perform policy iteration; see e.g. the least-squares policy iteration (LSPI) work of [Lagoudakis and Parr, 2002]. However, many of the same approaches can be directly translated to this LSPI task, and we will focus on the simpler task of policy evaluation.

4.1 Preliminaries

We again consider the standard RL framework introduced in Section 1.1. The value function we are interested in learning maps x to its long-term

expected value

$$V^\pi(x) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(x_t) \big| x_0 = x, \pi\big].$$

We can also define this quantity as the unique fixed-point of the Bellman operator $V^\pi = T^\pi V^\pi$, where the operator T^π is defined as

$$(T^\pi V)(x) = r(x) + \gamma \int_{\mathcal{X}} p(dx'|x, \pi(x)) V(x')$$

or more concisely as $T^\pi V = r + \gamma P^\pi V$, where P^π denotes the transition kernel induced by the policy and transition models. When both r and P^π are known, this quantity can be solved for analytically, however we will consider the situation that these quantities are either unknown, too large to evaluate, or both.

Rather than directly computing the value function, we will instead seek to approximate V^π with a space $\mathcal{F}$ consisting of linear combinations of k features $\phi : \mathcal{X} \to \mathbb{R}^k$ and weights $w \in \mathbb{R}^k$, i.e.

$$V^\pi(x) \approx \phi(x)^T w.$$

However, note that even for some function $V \in \mathcal{F}$ the result $T^\pi V$ of applying the Bellman operator to this value function will not necessarily lie in the span of the basis ϕ. Instead, we will adopt the approach of LSTD and approximate the resulting vector with the closest vector that *does* lie in $\mathcal{F}$. Let

$$\|f\|_\nu^2 = \int_{\mathcal{X}} f(x)^2 \nu(dx)$$

be the $\ell_2(\nu)$-norm of some function f with respect to the distribution ν. We can then introduce a projection operator Π such that

$$\begin{aligned} \Pi V(x) &= \phi(x)^T u^* \text{ where} \\ u^* &= \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - V\|_\nu^2. \end{aligned}$$

By combining the Bellman and projection operators, we can write the LSTD fixed-point as $\hat{V}^\pi = \Pi T^\pi \hat{V}^\pi$ which, for $\hat{V}^\pi = \phi^T w$, can be written as

$$w = u_w^* = \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - (r + \gamma P^\pi \phi^T w)\|_\nu^2. \tag{4.1}$$

Note that here we have written the result of the projection step u_w^* using the subscript to denote that it is dependant on w, exactly because this step corresponds to projecting the value function *given* by w. As an alternative to the explicit fixed-point approach given above, one can adopt the approach of [Antos et al., 2008] and write the value function as the solution $\hat{V}^\pi = \phi^T w^*$ to the following nested optimization problem

$$\begin{aligned} u_w^* &= \arg\min_{u \in \mathbb{R}^k} \|\phi^T u - (r + \gamma P^\pi \phi^T w)\|_\nu^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\phi^T w - \phi^T u_w^*\|_\nu^2. \end{aligned} \tag{4.2}$$

We will refer to the first problem as the *projection* step and the second as the *fixed-point* step. Note, that as it stands, $w^* = u_w^*$ is always a solution to the fixed point minimization. However, by expressing the fixed point in this way, we can in later sections of this chapter introduce methods for which we only approximately fulfill this fixed point by introducing regularization.

We should also emphasize that the fixed-point step is not the same as the projection of $\phi^T u_w^*$ since u_w^* itself depends on w, again as denoted by the subscript. If we let H denote the fixed-point operator that arises from the application of this second optimization problem then we can see that $\hat{V}^\pi = H(\Pi T^\pi \hat{V}^\pi)$. For the standard LSTD problem H is the identity, but we will introduce variations of this operator later for which this is not the case. Finally, this interpretation gives us a better picture of what is being optimized by LSTD. Given some value function $\hat{V}^\pi = \phi^T w$ we first apply the Bellman operator, which may take this function outside of the space $\mathcal{F}$ and then project this back down into the linear space. This is shown by the solid lines in Figure 4.1. The optimization problem of LSTD is then finding the value function that minimizes the distance between itself and

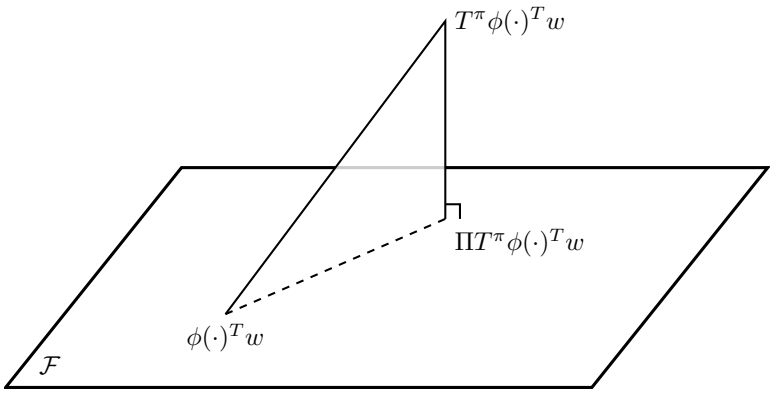

Figure 4.1: A graphical illustration of the LSTD problem. Here we see the Bellman operator which takes us out of the space $\mathcal{F}$ and the orthogonal projection back onto this space.

its projected Bellman image $\Pi T^\pi \hat{V}^\pi$ as denoted by the dashed line in this figure. See also [Antos et al., 2008] for more details.

Next, if we assume an n-length trajectory consisting of sampled transitions (x_i, a_i, r_i, x_i') from the MDP of interest, we can define the sample matrices

$$\Phi = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{bmatrix}, \; \Phi' = \begin{bmatrix} \phi(x_1')^T \\ \vdots \\ \phi(x_n')^T \end{bmatrix}, \; R = \begin{bmatrix} r(x_1) \\ \vdots \\ r(x_n) \end{bmatrix}.$$

We can then write an empirical version of (4.1) and solve the projection step as

$$\begin{aligned} u_w^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 \\ &= (\Phi^T \Phi)^{-1} \Phi^T (R + \gamma \Phi' w), \end{aligned} \tag{4.3}$$

and by setting $w = u_w^*$ can solve the fixed-point step,

$$\begin{aligned} w &= (\Phi^T (\Phi - \gamma \Phi'))^{-1} \Phi^T R \\ &= A^{-1} b. \end{aligned} \tag{4.4}$$

Here we have defined $A = \Phi^T(\Phi - \gamma\Phi')$ and $b = \Phi^T R$. While this solution provides an unbiased estimate of the value function, it can perform quite poorly when the number of samples is small in relation to the number of features. This type of scenario often results in *overfitting*, i.e. where we have more free parameters than observations, resulting in an overly complex model that is able to fit the noise of the system rather than the underlying system itself. In the next section, we examine various regularization methods designed to avoid overfitting.

4.2 Regularized LSTD

In this section we describe four different regularization methods which apply different penalty terms to the *projection* or to the *fixed-point* step introduced in (4.2). In the first two such schemes we do not penalize the fixed-point step, and we thus leave this step implicit. The final two schemes, however, rely on penalizing both sub-problems. Ultimately we describe a method which uses a mixture of ℓ_2 and ℓ_1 penalties, but that can be expressed as a standard Lasso problem.

4.2.1 ℓ_2 penalization (L_2)

The simplest form of regularization we can utilize involves adding an ℓ_2 penalty to the projection operator presented in (4.3), i.e.

$$\begin{aligned} u_w^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma\Phi' w)\|_2^2 + \beta\|u\|_2^2 \\ &= (\Phi^T\Phi + \beta I)^{-1}\Phi^T(R + \gamma\Phi' w). \end{aligned} \tag{4.5}$$

Similar to (4.4), we solve for the fixed point $w = u_w^*$ and obtain:

$$w = (\Phi^T(\Phi - \gamma\Phi') + \beta I)^{-1}\Phi^T R = (A + \beta I)^{-1} b. \tag{4.6}$$

We also see here the standard LSTD components A and b, the only difference with the original formulation being the addition of β along the diagonal of A. As we note later, we found that the use of an ℓ_2 penalty of this form was

primarily useful as a method of regularizing the projection matrix in order to avoid numerical instability.

4.2.2 ℓ_1 penalization (L_1)

We can also consider adopting an ℓ_1 penalty in the projection step, i.e.

$$u_w^* = \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_1. \tag{4.7}$$

The difficulty with this approach lies in the fact that there is now no closed-form solution to the optimization problem, a fact which causes difficulties when attempting to solve for the fixed-point $w = u_w^*$. Even though the projection step is just one of ℓ_1 penalized least-squares, the use of the fixed-point results in the problem not being equivalent to the Lasso. In fact, a more specialized algorithm is required to solve this problem. For a full description of this approach, and a related algorithm to solve this problem (LARSTD), we refer the reader to the work of [Kolter and Ng, 2009].

4.2.3 ℓ_2 and ℓ_2 penalization (L_{22})

Another approach we can take involves using the nested-optimization formulation of LSTD and applying regularization to both the projection and fixed-point steps. Such regularization was utilized by [Farahmand et al., 2009]. We can write this problem as

$$\begin{aligned} u_w^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_2^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\Phi w - \Phi u_w^*\|_2^2 + \beta' \|w\|_2^2. \end{aligned} \tag{4.8}$$

Just as we did in (4.5) we can find a closed-form solution to u_w^* and can then simplify the residual term of the fixed-point subproblem as

$$\Phi w - \Phi u_w^* = \Phi w - \underbrace{\Phi(\Phi^T \Phi + \beta I)^{-1} \Phi^T}_{\Sigma}(R + \gamma \Phi' w). \tag{4.9}$$

Here the matrix Σ represents the empirical ℓ_2 penalized projection operator, or *hat matrix*, which projects n-vectors onto the space spanned by the features Φ. We can then solve for w^* in closed form as

$$w^* = \arg\min_{w \in \mathbb{R}^k} \| \overbrace{(\Phi - \gamma \Sigma \Phi')}^{X} w - \overbrace{\Sigma R}^{y} \|_2^2 + \beta' \|w\|_2^2 = (X^T X + \beta' I)^{-1} X^T y. \tag{4.10}$$

We can also, however, formulate this problem in terms of the standard LSTD matrices (as defined in Section 4.1) by noting that for $C = \Phi(\Phi^T \Phi + \beta I)^{-1}$ we can write $X = C(A + \beta I)$ and $y = Cb$.

4.2.4 ℓ_2 and ℓ_1 penalization (L_{21})

Finally, we can also consider the same nested optimization problem as in in the previous scheme, but with an ℓ_1 penalty used in the fixed-point operator, i.e.

$$\begin{aligned} u_w^* &= \arg\min_{u \in \mathbb{R}^k} \|\Phi u - (R + \gamma \Phi' w)\|_2^2 + \beta \|u\|_2^2 \\ w^* &= \arg\min_{w \in \mathbb{R}^k} \|\Phi w - \Phi u_w^*\|_2^2 + \beta' \|w\|_1 \end{aligned} \tag{4.11}$$

Here we can again use the simplification from (4.10) to write the solution as

$$w^* = \arg\min_{w \in \mathbb{R}^k} \| \underbrace{(\Phi - \gamma \Sigma \Phi')}_{X} w - \underbrace{\Sigma R}_{y} \|_2^2 + \beta' \|w\|_1. \tag{4.12}$$

As a result we have now transformed L_{21} into a standard Lasso problem, in terms of X and y, to which we can apply any off-the-shelf solution method. We note that the use of the ℓ_2 penalty primarily provides for the avoidance of numerical stability in the projection step. Alternatively one could also consider using a mixture of ℓ_1 and ℓ_2 penalization in the fixed-point step as in [Zou and Hastie, 2005], although we do not consider this penalization scheme here.

4.3 Standardizing the data

In standard applications of the Lasso, it is often assumed that the feature matrix has columns that are standardized (i.e. that they are centered and zero-mean) and that the response vector is centered. Although we will briefly discuss the reasons for this, a more comprehensive treatment is given in e.g. [Friedman et al., 2001, Section 3.4]. The centering is assumed because we generally want to estimate an unpenalized bias term w_0 so as to avoid making the problem dependent on the responses' origin. In doing so, the bias is given by the mean response and the remaining weights w can then be estimated using no bias term and centering the features and responses. The scaling of the features is perhaps more important and we can first note that the Lasso estimate is not invariant to this scaling. The scaling essentially evens the playing field for deciding which features are important, but it can also greatly impact the convergence speed of solution methods [Friedman et al., 2001]. In this section we will now describe how to incorporate these assumptions into the L_{21} scheme introduced in the previous section.

We will now explicitly introduce a bias term w_0 into the value function approximation with $V^\pi(x) \approx \phi(x)^T w + w_0$. We can then rewrite the nested optimization problem as

$$(u_w^*, u_{w0}^*) = \arg\min_{u,u_0} \|\Phi u + u_0 - (R + \gamma(\Phi' w + w_0))\|_2^2 + \beta_2 \|u\|_2^2 \tag{4.13}$$

$$(w^*, w_0^*) = \arg\min_{w,w_0} \|\Phi w + w_0 - (\Phi u_w^* + u_{w0}^*)\|_2^2 + \beta_1 \|w\|_1. \tag{4.14}$$

Here we have also introduced a bias term u_{w0}^* corresponding to the resulting bias of the projection step. Although this notation is somewhat cumbersome, it has been chosen to most closely match the notation used in introducing the original penalized problem in the abvoe sections.

Now, before solving these problems we will first introduce the following notation: $\bar{\Phi} = \text{mean}(\Phi)$ is the row-vector of feature means, $\tilde{\Phi} = \Phi - \mathbf{1}_n \bar{\Phi}$ are the centered features where $\mathbf{1}_n$ is a column vector of n ones, and $\hat{\Phi} = \tilde{\Phi}\Omega$ consists of the centered and rescaled features given a scaling matrix Ω whose diagonal elements consist of the inverse standard deviations of the feature

matrix Φ. Similar terms are introduced for centering both Φ' and R.

For both the projection and fixed-point sub-problems we can solve for the bias and weight terms individually. In both cases the bias is given by the mean response minus the mean of the features and the optimal weights. For the bias of the projection step this can be written as

$$u^*_{w0} = (\bar{R} + \gamma\bar{\Phi}'w + \gamma w_0) - \bar{\Phi}u^*_w. \tag{4.15}$$

The bias now depends upon finding u^*_w, but we can solve for this by centering both the features and responses and solving the following problem:

$$\Omega^{-1}u^*_w = \arg\min_{u\in\mathbb{R}^k} \|\hat{\Phi}u - (\tilde{R} + \gamma\tilde{\Phi}'w)\|_2^2 + \beta_2\|u\|_2^2 \tag{4.16}$$

$$= (\hat{\Phi}^T\hat{\Phi} + \beta_2)^{-1}\hat{\Phi}^T(\tilde{R} + \gamma\tilde{\Phi}'w). \tag{4.17}$$

Note, however, that since we are solving this minimization problem using a scaled feature matrix (i.e. $\hat{\Phi}$) we must remember to rescale back into the original units, which accounts for the the inverse of Ω.

We can now write the projected value function as

$$\begin{aligned}\Phi u^*_w + u^*_{w0} &= (\Phi - \mathbf{1}_n\bar{\Phi})u^* + (\bar{R} + \gamma\bar{\Phi}'w + \gamma w_0) \\ &= \underbrace{\hat{\Phi}(\hat{\Phi}^T\hat{\Phi} + \beta_2)^{-1}\hat{\Phi}^T}_{\Sigma}(\tilde{R} + \gamma\tilde{\Phi}'w) + (\bar{R} + \gamma\bar{\Phi}'w + \gamma w_0).\end{aligned}$$

Here we have combined terms using the definitions introduced earlier and we can see the Σ term is the ℓ_2 penalized projection matrix onto the *centered and rescaled* features. Plugging this projected value function into the fixed-point problem we arrive at

$$\begin{aligned}&(w^*, w^*_0) \\ &= \arg\min_{w,w_0} \|\Phi w + w_0 - \Phi u^*_w - u^*_{w0}\|_2^2 + \beta_1\|w\|_1 \\ &= \arg\min_{w,w_0} \|\underbrace{(\Phi - \gamma\Sigma\tilde{\Phi}' - \gamma\mathbf{1}_n\bar{\Phi}')}_{X} w + (1-\gamma)w_0 - \underbrace{(\Sigma\tilde{R} + \bar{R})}_{y}\|_2^2 + \beta_1\|w\|_1.\end{aligned}$$

We can see then that this is very closely related to the original formulation from (4.12), however here we are projecting according to the centered/rescaled features and X and y have additional components related to the mean next-state features and mean rewards respectively.

Now we truly have a standard Lasso problem. We can solve for the optimum w^* using any off-the-shelf Lasso solver using the scaled/centered matrix X and centered vector y (and again if we rescale X we must return the output to the original scaling). Given the optimal weights we can then solve for the bias term as $(1-\gamma)w_0^* = \bar{y} - \bar{X}w^*$, where these terms again denote the mean response and the mean of the features respectively.

4.4 Discussion of the different regularization schemes

In this work, we are particularly interested in the situation where the number of features is greater than the number of samples, $k > n$. Drawing on results from standard regression problems, we would expect the ℓ_2-based methods to perform poorly in this situation. In fact, we will see in the later experiments that just such a drop-off in performance occurs at the point when k overtakes n. We would, however, expect the ℓ_1-based methods, due to the feature selection properties of this penalty, to continue performing well so long as there is some small subset of relevant features that fit the value function well. In the setting of regression this behavior is well-established both empirically and theoretically [e.g., Bunea et al., 2007, Tibshirani, 1996]. The extension of this behavior to the RL setting, although expected, is not entirely straightforward due to the fixed-point aspect of the learning process. In the rest of this section, we will discuss and contrast how the ℓ_1 regularization is implemented in the two penalized methods, L_1 and L_{21}.

The main difference between the two schemes, L_1 and L_{21}, lies in the choice of where to place the ℓ_1 penalty. The L_1 approach uses a penalty directly in the projection operator. This is a straight-forward modification of the projection, and we would expect that, if the result of applying the Bellman operator can be well-represented by a sparse subset of features, the

projection will find this. In fact, there are some recent theoretical guarantees that, if the target value function V^π is sparse, the L_1 scheme will be able to take advantage of this fact as shown by [Ghavamzadeh et al., 2011]. More precisely, if $s \ll k$ is the number of features needed to represent V^π, the prediction error of L_1 directly scales with s instead of k as shown by Ghavamzadeh et al.. This suggests that L_1 will perform well until the number of samples is bigger than the number of relevant features, $n \geq s$. However, we must note that when the projection step is combined with the fixed point, it is not entirely clear what is being optimized by the L_1 procedure. In fact, [Kolter and Ng, 2009] claim that this approach does not correspond to any optimization problem in w. Further, from an algorithmic point of view, since the fixed-point is applied *after* the ℓ_1 optimization, this approach does not correspond to a Lasso problem, resulting in the need to use more specialized algorithms to solve for w.

Alternatively, the L_{21} approach places an ℓ_1 penalty in the fixed-point step. The main benefit of this approach is that it allows us to cast the regularized LSTD problem as a Lasso problem, to which we can apply general-purpose Lasso or ℓ_1 solvers. This also allows us to more straightforwardly apply results such as those of Section 4.3, something that is not easily done in the L_1 scheme. The application of standardization to the feature matrix has a great deal of impact on the results of Lasso in regression problems and we would expect it to have a great deal of impact for RL as well (in the later experiments we will see some evidence of this). We also found that the ability to standardize the features played a role in the previously mentioned flexibility of L_{21}. In fact, we found that without standardization of the features it was difficult to apply certain iterative methods such as those discussed in [Friedman et al., 2007, Schmidt, 2010] due to slow convergence.

One potential downside to the L_{21} approach is the necessity of using an ℓ_2 penalty in the projection step. It is not entirely clear what effect this has on the resulting algorithm, although in our preliminary experiments we found that this penalty was primarily useful in computing the projection matrix Σ, and making sure that the required matrix was non-singular. Further, the necessity of computing Σ does make L_{21} somewhat more expensive than

L_1, however as this matrix only depends on Φ, using this procedure inside of a policy iteration scheme would only require the matrix being computed once. Finally, while there does exist some theoretical evidence that the L_1 approach is able to take advantage of the sparsity of V^π, no such guarantees exist yet for L_{21}. The combination of penalizing both the projection and fixed-point steps introduce this theoretical difficulty, however see for example the work of [Farahmand et al., 2009] for non-sparse guarantees in the ℓ_2 penalized setting. Our experiments do, however, seem to indicate that L_{21} is able to capitalize on such value functions, and may even perform better when the value function is "truly sparse".

4.5 Experimental results

In comparing the performance of the regularization schemes introduced in Section 4.2, we will consider the *chain problem* introduced in [Lagoudakis and Parr, 2002]. In these experiments we will utilize a 20-state, 2-action MDP wherein states are connected in a chain and where upon taking action *left* from state x the system transitions to state $x-1$ with probability p and $x+1$ with probability $1-p$. The same holds for action *right* but in reverse and at both ends of the chain a successful right (or left) action leaves the state unchanged. We will restrict ourselves to the problem of policy evaluation and will evaluate the performance of the regularization schemes as the relative number of features k (as compared to the number of samples n) is varied. In particular, we will consider $k = s + \bar{s}$ features consisting of s "relevant" features including a single constant feature and some number of radial-basis functions (RBFs) spread uniformly over the state space. The additional features consist of "irrelevant" or "noise" features implemented by extending the state-space of the chain model to include $\bar{s}$ additional dimensions where each additional dimension is independently distributed $x_t^{i+1} \sim \mathcal{N}(0, \sigma^2)$ for all time indices t. The value of each noise feature is then given by the corresponding state's value, i.e. $\phi_{s+i}(x_t) = x_t^{i+1}$ for $1 \leq i \leq \bar{s}$. These additional features can also be thought of as *random* features of the standard chain model.

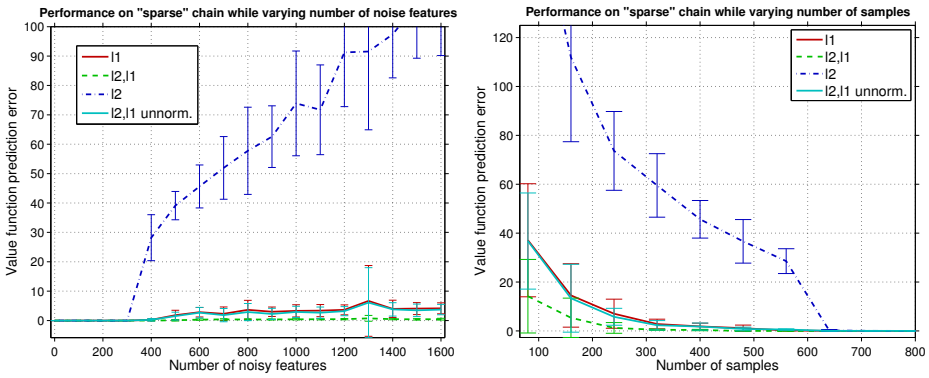

Figure 4.2: Performance of policy evaluation on a sparse value function.

We will first consider the case where the value function can be constructed as a sparse combination of features. In order to test this scenario, we start with a sparse value function and work backwards. We will consider a value function given as a linear combination of s relevant features, i.e. $V(x) = \sum_{i=1}^{s} \phi_i(x) w_i^*$ and we then define the reward of our model as $r(x, x') = V(x) - \gamma V(x')$. We have, in this case, artificially constructed our reward in order to enforce the desired sparse value function. In this experiment we used $s = 10$ and varied the number irrelevant features. In each of 20 runs we sample n samples on-policy and use them to approximate the value function. Here we used a policy π which takes action *left* on the first 10 states and action *right* on the next 10, although due to our construction of the reward function for this problem, the actual policy used does not affect the value function. The resulting approximation was then evaluated at 500 test points and the empirical error between this and the true value function is computed. The policy is kept fixed across all runs and within each run we perform cross-validation over the regularization parameter.

In Figure 4.2 we compare the L_2, L_1, and L_{21} variants described in Section 4.2; here we omit the L_{22} variant to avoid clutter[1]. L_{21} can be run using any Lasso solver. In our implementation we both tested LARS [Efron

[1]Preliminary experimental results seemed to show that the L_{22} variant performed similarly to the L_2 scheme.

et al., 2004] and a number of gradient descent procedures [Friedman et al., 2007, Schmidt, 2010]. In order to generate the value function we sampled our true weights w_i^* uniformly in the range $[-5, 5]$ for every run. From our experiments, however, we saw that the true values of w did not play a significant role in explaining the error of each method: this variance was more attributable to the data samples generated. The first of these plots shows the behavior using $n = 400$ samples (consisting of 20 episodes of length 20) while varying the number of noise features. We can first note that all algorithms perform quite well until the number of features k approaches the number of samples n, but after around 400 noise features the algorithms begin to differentiate. L_2 immediately starts overfitting the data and its prediction error grows almost linearly with the total number of features k. On the other hand, the ℓ_1 regularization approaches are expected to depend more directly on the *true* dimensionality s rather than k. In fact, we can see that both L_1 and L_{21} are not drastically affected by overfitting when k becomes greater than n. In the plot, we also include the performance of L_{21} without the rescaling described in Section 4.3 and here this performs about as well as the L_1 approach (using LARSTD). The rescaled version, however, is much better able to deal with the increase in number of noise features as can be seen by its very low prediction error, and in comparison to the other methods this is nearly constant. The second plot shows the results of $\bar{s} = 600$ noise features and varying the number of samples n, and we see similar differences in performance.

Next we consider the standard chain problem which uses a reward of 1 when the relevant state (x_1) is at either endpoint and 0 elsewhere. In the first of these experiments we use $s = 6$ relevant features and again vary the number irrelevant features. In each of 20 runs we again sample 400 data points (consisting of 20 episodes of length 20) on-policy (using the optimal policy) and use these points to approximate the value function. Given this model we can compute the true value function and in order to generate the plots we evaluate the empirical error between this and the estimated value function at 500 test points. Again, the policy is kept fixed and within each run we perform cross-validation over the regularization parameter.

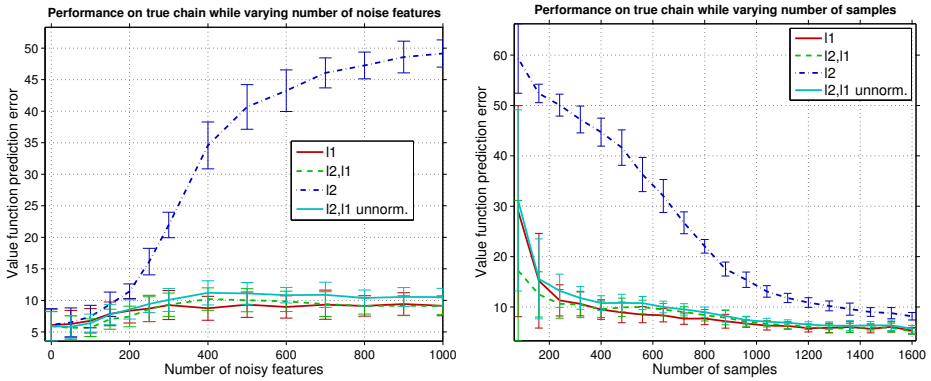

Figure 4.3: Performance of policy evaluation on the chain model for a fixed policy.

The results of Figure 4.3 closely echo the results of Figure 4.2. Here, however, we can see that there is a somewhat more significant difference between the unscaled version of L_{21} and the L_1 variant, although this performance is still significantly better than L_2 alone. We also see that the prediction error is no longer zero (as would be expected due to the fact that now the value function can be exactly represented in the linear space). We can also see, however, that there is a more significant increase in this error as we approach the point where the number of features equals the number of samples. Again, the second plot shows the results of $\bar{s} = 600$ noise features while varying the number of samples n, and again we see similar differences in performance.

In Figure 4.4 we show the effect of the number of irrelevant features on the best penalty parameters chosen via cross-validation on both the sparse chain and the standard chain model. The ℓ_1 based methods use LARS (or the LARSTD modification) to find the regularization path and for the ℓ_2 based method we used a grid of 20 parameters logarithmically spaced between 10^{-6} and 10. In both figures we show results where the number of samples is fixed at $n = 400$ and the number of noisy features is increased. The main point to see here is the much greater variability in this parameter value for the L_2 method. The L_1 method then may also be more variable

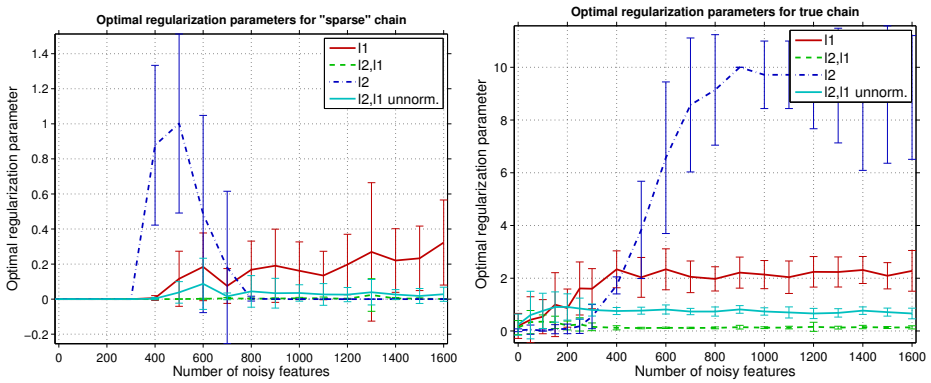

Figure 4.4: Effect of the number of irrelevant features on the optimal penalty parameters chosen via cross-validation.

than the L_{21} method, but this level of increase could also be attributable to different scaling in this parameter. For these experiments we also found that the ℓ_2 parameter of L_{21} was primarily useful in ensuring the relevant matrix Σ was nonsingular, and as a result we were able to set this parameter to a quite low-value 10^{-6} and in these plots we only show the ℓ_1 penalty. Based on this it would seem that the tuning of the penalty parameter for L_{21} is at least *no harder* than that of L_1 due to the minimal effect of the second parameter Finally, we point out that the decline in the penalty parameter of L_2 for the sparse chain is attributable to the fact that after approximately 600–800 noise features all parameters were "equally bad".

Finally, we analyzed the effect of the number irrelevant features on the number of iterations when solving the problem using L_1/LARSTD and when solving the L_{21} problem using LARS [Efron et al., 2004]. Although these results are not entirely conclusive, it appears that the L_{21} approach may require fewer iterations for sparse models, but more for non-sparse models. As mentioned earlier, the additional flexibility of the L_{21} scheme allows us to apply more sophisticated optimizations schemes such as the coordinate descent approach of [Friedman et al., 2007] or other iterative schemes such as those of [Schmidt, 2010]. We applied both of these procedures, code for which is available online and is quite fast, to our problem. However, the

comparison of the complexity of these different algorithms is not simple and we leave it for future work.

4.6 Chapter summary and conclusions

In this chapter we presented a number of regularization schemes for LSTD, culminating in a novel approach using nested ℓ_2 and ℓ_1 penalties. We then showed how to apply standardization techniques from the regression literature to this problem and discussed preliminary experiments comparing this approach to other schemes based on ℓ_2 and ℓ_1 penalized projection. The main benefit of this approach is the additional flexibility it provides over ℓ_1 projected LSTD approaches (such as LARSTD), and the ability to use standard Lasso solvers and techniques. This means that any new solver for Lasso could be immediately used in L_{21}, whereas this would need a rethinking of the whole algorithm for L_1. We should also note a similar approach, derived in parallel, of [Geist and Scherrer, 2011].

There are also a number of open areas of research with this approach. A more thorough experimental analysis is needed, including an extension of this method to the problem of policy iteration via LSPI. A theoretical analysis of this approach is also called for, in line with the previously noted analysis of the L_1 scheme presented in [Ghavamzadeh et al., 2011]. Finally, a better understanding of the ramifications of including the ℓ_2 penalty would be most beneficial.

Chapter 5

Multi-armed bandits

> *the problem is a classic one; it was formulated during the war, and efforts to solve it 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.*
>
> –Peter Whittle, on multi-armed bandits

In this chapter we will consider a model that is in some sense a simplification of the MDP problem presented in previous chapters. For the moment we will abandon the notion of a state space and consider a model consisting only of an action space $\mathcal{A}$. At each time n the decision maker chooses an action $a_n \in \mathcal{A}$ and receives reward r_n from the environment. More concretely, we will assume this interaction proceeds as follows:

For each round $n = 1, 2, \ldots$

1. the decision maker chooses an action $a_n \in \mathcal{A}$;
2. the decision maker receives reward r_n.

This model is known in the literature as a multi-armed bandit problem. Here the actions are commonly described as *arms* in an analogy to the problem faced by a gambler when deciding which of several slot machines (or *one-armed bandits*) to play. The gambler must sequentially decide which

of these arms to pull and he or she will receive some possibly random payout for pulling the ath arm. In this chapter, following the classical formulation of [Robbins, 1952], we will assume that rewards for each arm are independently distributed $R^{(a)} \sim g_a(\cdot)$ according to some fixed but unknown distribution. Throughout this chapter we will let R_n denote the random variable associated with the reward at time n and let $R_n^{(a)}$ denote the reward had we instead chosen action a. Later, in Chapter 6, we will consider correlated rewards, however note that in this chapter we will assume independent reward distributions for each arm.

At round n the decision-maker will have interacted with the bandit process and obtained data concerning the rewards for each arm of the form $\mathcal{D}_{n-1} = \big(a_{1:n-1}, r_{1:n-1}\big)$. We can then use this information about past rewards to inform the decision of which arm a_n to select at time n. Throughout the rest of this chapter we will make use of *index* strategies by way of quantities $\nu_{na} : (\mathcal{A} \times \mathbb{R})^{n-1} \to \mathbb{R}$ which at each round n map the past data to a scalar index quantity. The action selection strategy is then completely determined by this index, $a_n = \arg\max_{a \in \mathcal{A}} \nu_{na}(\mathcal{D}_{n-1})$. We will also generally write this index quantity more simply as ν_{na} and leave the dependence on $\mathcal{D}_{n-1}$ implicit except where necessary. Pseudo-code for arm selection using an index strategy is shown in Algorithm 2. While the definition of an index strategy is somewhat opaque, intuitively this index corresponds to the *value* of selecting the ath arm given data $\mathcal{D}_{n-1}$. For example, we could consider a strategy that tries every arm once, and afterwards follows a strategy such that ν_{na} is the empirical mean of arm a. However this strategy is in some sense *too greedy* as it does not take into account the uncertainty inherent in the arms, i.e. that some of the arms that have been pulled less often may have higher rewards that we just haven't seen yet. As we will see in the next section, this intuition can be made much more concrete.

Finally, we should note a few details about the index computation itself. First, we have written the index in such a way that it can depend on n, the current stage of the decision process. This is important as many of the strategies we will introduce in this chapter will either directly or indirectly depend on this quantity in order to adjust their exploration as time goes on.

Algorithm 2
Bandit arm selection using a general index policy.

1: Initialize the index quantity ν_{1a}.
2: **for** $n = 1, 2, \ldots$ **do**
3: Select action $a_n = \arg\max_a \nu_{na}$ and receive reward r_n.
4: Update the data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup (a_n, r_n)$.
5: Compute the index quantities ν_{na}.
6: **end for**

We have also written this function in such a way that the index quantity associated with arm a is also allowed to depend on the data acquired for every other arm. Many of the strategies we introduce can also be written such that the index ν_{na} only depends on the subset of data $\mathcal{D}_{n-1}$ such that $a_t = a$, i.e. these indices are independent. For example, the greedy strategy introduced above only makes use of independent indices. However, this generalization allows for index computations which directly compare the value of different arms.

The rest of this chapter serves as a review of the relevant literature, with empirical comparisons given of the various methods we will introduce. In the Section 5.1 of this chapter we will introduce the optimal Bayesian approach of [Gittins, 1979]. In particular, we will see that the computations necessary to obtain the optimal policy for this problem can be formulated as an MDP in order to relate to the methods of the previous chapters. This procedure can, however, prove quite expensive in practice. Instead, in Sections 5.2–5.3 we will introduce a number of alternative index policies from the literature the computation of which are much more amenable. We will also introduce the concept of *cumulative regret* as a measure of the performance of these algorithms and detail their approximation guarantees. We will also empirically compare these methods on a set of test problems.

The policies introduced in Section 5.2 are quite useful when the decision maker wants to optimize the online performance of the decision making process—i.e. the sum of (possibly discounted) rewards both now and in the future. Alternatively we are often faced with the problem of identifying the single best arm. This often comes about when the arms correspond to an

optimization problem which must be solved with limited resources. Consider, for example, the problem of drug trials, where each arm corresponds to a possible drug combination and each round corresponds to a single experiment with a particular combination. At the end of the trial we are not interested in how well the bandit process performed during the testing, we are only concerned with the performance of the best drug combination. In Sections 5.4–5.5 we will introduce modifications to the policies of Section 5.2 from the literature that are able to take advantage of this changed objective. In particular we will introduce the notion of *simple regret* which captures this notion of only being concerned with later performance, and distinguish it from *cumulative regret*. We will also study the empirical performance of these methods. Finally, these sections as a whole will serve as a branch to the topic of Chapter 6 wherein we will consider a similar bandit process but with correlated, and generally continuous, arms.

5.1 The optimal Bayesian solution

We will first consider the standard task of constructing a Bayesian model for a single arm. Let $p(r|\theta)$ denote the probability of observing rewards r conditioned on some unknown parameters θ and let $p(\theta)$ denote these parameters' prior density. Given this model and n reward samples $r_{1:n}$ we can write the posterior density of the parameters as

$$p(\theta|r_{1:n}) \propto p(\theta) \prod_{i=1}^{n} p(r_i|\theta). \tag{5.1}$$

We can easily extend this formulation to a collection of independent arms $a \in \mathcal{A}$ where we must now make use of the previously introduced collection of data $\mathcal{D}_n = \big(a_{1:n}, r_{1:n}\big)$. We will next assume likelihood and prior models $p(r|\theta_a)$ and $p(\theta_a)$ associated with each arm a. The posterior for the ath arm, which we will write as $p(\theta_a|\mathcal{D}_n)$, is then given exactly as above by considering only those rewards $(r_t, \dots)$ such that $a_t = a$; this is due to the fact that we are assuming *independent* arms. Note that as a result, selecting arm a will only change the posterior for that particular arm—a fact that

will have implications later in this section. Finally, we can also write the posterior reward predictions as

$$p(r|a, \mathcal{D}_n) = \int p(r|\theta_a)\, p(\theta_a|\mathcal{D}_n)\, d\theta_a, \tag{5.2}$$

i.e. this is a distribution over the reward r that would be obtained from selecting arm a, predicted using the data from n previous arm pulls. Below we illustrate this Bayesian model using an arm with Bernoulli reward likelihood and a Beta prior. We will use this as a running example throughout the rest of this chapter.

Example. Consider the problem of a single arm with Bernoulli distributed rewards with success probability given by θ. We will also assume a prior density given by $p(\theta) = \text{Beta}(\theta; \alpha_0, \beta_0)$. Given this conjugate prior we can then see that the posterior is also a Beta distribution

$$p(\theta|\mathcal{D}_n) = \text{Beta}(\theta; \alpha_n, \beta_n),$$

where $\alpha_n = \alpha_0 + \mathbb{I}_1(r_1) + \cdots + \mathbb{I}_1(r_n)$ is the number of successes in n trials (plus the pseudo-count α_0) and β_n is defined similarly for the number of failures. Finally, the posterior reward distribution is Beta-Bernoulli with parameters given by (α_n, β_n) and as a result has mean and variance given by

$$\begin{aligned}\mathbb{E}[R|\mathcal{D}_n] &= \alpha_n \big/ (\alpha_n + \beta_n), \\ \text{Var}[R|\mathcal{D}_n] &= \alpha_n \beta_n \big/ (\alpha_n + \beta_n)^2.\end{aligned}$$

With this background in place we will now turn to the problem of how to select the next action in a multi-armed bandit, for which we must introduce an index strategy ν_{na}. One possible strategy for selecting the next action as alluded to previously is simply to choose the arm with the highest expected reward under the posterior distribution, i.e. $\nu_{na} = \mathbb{E}[R_n|a, \mathcal{D}_{n-1}]$. This myopic strategy, unfortunately, is sub-optimal. Consider, for example, two

Bernoulli arms with Beta priors whose current sufficient statistics are given by $\alpha_1 = \beta_1 = 1$ and $\alpha_2 = \beta_2 = 10$. The expected reward for both arms is 0.5, however the posterior associated with the first arm has much higher variance. It is therefore much more likely that pulling arm one will result in significant changes to the posterior distribution than it would be to see comparable changes upon pulling arm two. Intuitively this makes pulling arm one a much more attractive option, as both arms have the same expected immediate reward, but by pulling arm one we can gain much more information about the actual arm distribution. Somewhat more concretely, we can see that there is a much higher probability that the true mean of arm one is greater than its posterior mean.

Rather than appealing to intuition as a rule for which arm to select, we can instead formalize the problem of choosing arms in a multi-armed bandit as an MDP. Note, however, that the MDP we will introduce is *separate* from the underlying reward function of the bandit itself—this quantity is unknown to the decision maker. Instead we will introduce an MDP such that by solving for the value function of this MDP we will obtain the expected value of taking each action $a \in \mathcal{A}$, conditioned on the current data $\mathcal{D}_{n-1}$. This in turn will allow us to compute the index quantity $\nu_{na}(\mathcal{D}_{n-1})$. In the next two sub-sections we will first introduce the MDP that we are interested in solving, and then we will show an efficient method for solving for the value of each action.

5.1.1 Modeling the bandit problem as an MDP

We can now turn to the specification of our MDP model, the action space of which can easily be seen to correspond to the set of k arms $\mathcal{A}$. We will now assume that each arm has the same prior and likelihood, and we will assume a set of sufficient statistics $\mathcal{X}$ that fully describe the posterior for each arm. In particular, we will let $x_{na} \in \mathcal{X}$ denote the sufficient statistics for the ath arm at the beginning of round n, from which we can write the posterior over parameters and rewards as

$$p(\theta_a|\mathcal{D}_{n-1}) = p(\theta_a|x_{na}),$$

$$p(r|a, \mathcal{D}_{n-1}) = p(r|x_{na}).$$

The state space of our MDP is then given by $\mathcal{X}^k$, i.e. the collection of sufficient statistics for each arm. The assumption of sufficient statistics means that any posterior $p(\theta_a|\mathcal{D}_{n-1})$ can be represented exactly by some statistics x_{na}; i.e. we can really think of our state as *being* the posterior distribution for each arm, but that we represent this distribution using its sufficient statistics. Alternatively, we could have instead modeled the state of this process using the data tuple $\mathcal{D}_{n-1}$, however that would mean that we would have a state of growing dimension. Further, the assumption of sufficient statistics is not overly arduous, as it will often hold in the problems we are interested in. For example, the Beta-Bernoulli example introduced above has statistics given by $x_{na} = (\alpha_{na}, \beta_{na})$, i.e. the success/failure counts for each arm.

As an aside, we should also note that the state space of this MDP is not an actual, physical state but is instead a so-called *informational* state denoting a posterior distribution over the "true" model parameters. In fact, we can view this problem as a simple *partially-observable* MDP, see e.g. [Cassandra et al., 1995], where the latent state is fixed. In this setting the underlying, hidden state of the POMDP corresponds with the true model parameters. However, POMDPs are often solved by transforming them into MDPs whose states are given by the posterior statistics, exactly as we will do in this section.

Next we will assume the existence of a function $f : \mathcal{X} \times \mathbb{R} \to \mathcal{X}$, which we will use, merely to formalize the posterior update introduced at the beginning of this section. In order to explain this function, consider a bandit process in state x_n, from which the decision maker takes action a and receives rewards r. Then the next state of the process will be of the form $x_{n+1,a'} = x_{na'}$ for actions $a' \neq a$ and $x_{n+1,a} = f(x_n, r)$. In other words f is used to update the statistics of the selected arm, whereas all other arms stay the same. As an example, for the Beta-Bernoulli model introduced above this

update corresponds to

$$f((\alpha, \beta), r) = (\alpha + \mathbb{I}_1(r), \beta + \mathbb{I}_0(r)),$$

where we need only update the success/failure counts of the selected arm. Note that the fact that taking action a only changes the statistics corresponding to that arm follows from the independence of each arm's reward distribution. In other words, the posterior for arm a only changes once we acquire more data about that arm. We should also note that this function describes the deterministic transition model, *given* the rewards. In other words, all the stochasticity in the transitions are determined by these rewards.

We can now introduce stochastic rewards for this MDP given exactly by the posterior predictive distribution over rewards. Upon taking action a given the current statistics x_n, rewards are distributed according to $p(r|x_{na})$. Note again that this reward model is separate from the underlying, unknown rewards g_a. In fact, we can think of this probability distribution as a surrogate for the unknown rewards that allows us to plan into the future without performing any actual arm pulls. And again, we note that this posterior distribution is independent for each arm. Later, in Chapter 6 we will introduce similar approaches which make use of a Gaussian process surrogate in order to take into account correlated arms. Finally, by taking expectations with respect to this reward model we can compute the expected reward for pulling each arm, and by extension the expected effects that these reward observations would have on the posterior. Again, we emphasize that as a result the reward and transition models are based only on the posterior distribution of each arm, and are wholly separate from the underlying reward distributions g_a precisely because these distributions are unknown to the decision maker.

This formulation of the problem follows from the work of [Bellman, 1956] for two-armed bandits and [Gittins and Jones, 1979]; see also the extensive coverage of [Gittins et al., 2011]. We should now note what using this MDP means. Consider a bandit process at round n which has observed data

$\mathcal{D}_{n-1}$. We can map this observed data to a state x_{na} for each arm and use the MDP model that we have just introduced to extrapolate into the future the expected rewards that would be gained on pulling each arm. And again, these expectations are computed based purely on the posterior distribution induced by the data $\mathcal{D}_{n-1}$. One question that remains is *why* we need the MDP formulation when we already have the posterior distribution at hand? The answer to this question is that by pulling some arm a we will gain not only rewards for pulling that arm, but additional information that we can incorporate into future posteriors. As a result we need the MDP in order to take into account the future rewards, but also the affect these future rewards will have on future posteriors.

Now, at first glance, the actual computation involved with solving this MDP seems daunting. Suppose we were to approximate the value of taking some action by computing expected rewards N steps into the future. This expectation would require integrating over all k^N possible sequences of actions, which scales exponentially with the number of arms. Surprisingly, the optimal policy for this class of problems takes the form of an *independent* index depending only on the state of each arm x_{na}. This fact relies on the independence of each individual arm, and we will introduce this method in the next sub-section. This optimal index strategy was originally introduced by [Gittins, 1979] under the name *dynamic allocation indices*, and is now more commonly known as the *Gittins index*. The method we will detail for computing this process is known as the *calibration* method, and its exposition is loosely based on that of [Gittins et al., 2011].

5.1.2 Computing the Gittins index

In order to introduce Gittins' index policy we will first turn to the problem of a "one-armed" bandit. In this problem we consider a single bandit arm which starts in state x_1 and a deterministic reference arm with no state which when pulled always gives known reward λ. Suppose now that the bandit arm is chosen at time $n = 1$ and an optimal policy is followed for all subsequent steps. Let τ be the first time step at which the reference arm

is chosen. Once this time step is reached the posterior distribution for the bandit arm will not change, i.e. $x_\tau = x_{\tau+1}$, since new observations about an arm only occur when the arm is chosen. As a result, if it is optimal to select the reference arm at time τ it will remain optimal for all following steps. In other words, optimizing the policy of this problem reduces to finding the first step $\tau > 1$ at which we will switch to the reference arm. Equivalently, by optimizing over τ we are finding the maximum number of times under expectation that we should pull the bandit arm after pulling it at least once. By directly optimizing over this time horizon τ we can write the value of pulling the bandit arm at least once from state x_1 as

$$\begin{aligned} V(x_1, \lambda) &= \sup_{\tau>1} \left\{ \mathbb{E}\left[\sum_{n=1}^{\tau-1} \gamma^{n-1} R_n \middle| x_1 \right] + \gamma^{\tau-1}(1 + \gamma + \gamma^2 + \cdots)\lambda \right\} \\ &= \sup_{\tau>1} \left\{ \mathbb{E}\left[\sum_{n=1}^{\tau-1} \gamma^{n-1} R_n \middle| x_1 \right] + \gamma^{\tau-1} \frac{\lambda}{1-\gamma} \right\}. \end{aligned}$$

The second term inside the supremum corresponds to the discounted rewards accrued once we start pulling the reference arm, where the leading $\gamma^{\tau-1}$ term follows from the initial discount of waiting τ time steps. Note also that the fractional value $1/(1-\gamma)$ follows from simplifying the geometric series $1 + \gamma + \cdots$. As a consequence, we can easily see that were we to pull the reference arm at all times we would accrue rewards $\lambda/(1-\gamma)$ due to the constant reward of this arm.

Let us now consider the *advantage* of the bandit arm over the reference arm, which we can obtain by subtracting the long-term value of always pulling the reference arm from $V(x_1, \lambda)$. We will write this quantity as

$$\begin{aligned} D(x_1, \lambda) &= V(x_1, \lambda) - \frac{\lambda}{1-\gamma} \\ &= \sup_{\tau>1} \left\{ \mathbb{E}\left[\sum_{n=1}^{\tau-1} \gamma^{n-1} R_n \middle| x_1 \right] - (1 - \gamma^{\tau-1}) \frac{\lambda}{1-\gamma} \right\} \qquad (5.3) \\ &= \sup_{\tau>1} \mathbb{E}\left[\sum_{n=1}^{\tau-1} \gamma^{n-1} (R_n - \lambda) \middle| x_1 \right]. \qquad (5.4) \end{aligned}$$

The final equality follows by expanding $\lambda/(1-\gamma)$ back to its geometric series, or more simply by noticing that the advantage of the bandit arm can also be seen as the expected difference at each time step up until time τ. Once time τ is reached the bandit arm offers no advantage since we will no longer be pulling that arm. By setting $D(x_1, \lambda)$ equal to zero and solving for λ we can find the maximum value of the reference arm under which we would be indifferent between the reference arm and the bandit arm. We can see from (5.3) that for any fixed τ this quantity is a decreasing linear function of λ, from which the supremum over τ must be convex and decreasing. As a result for any x_1 the root of $D(x_1, \lambda)$ must exist and be unique. Intuitively this value of λ represents the maximum cost that we would pay in order to pull an arm with statistics given by x_1, or in other words the value of this arm.

Returning to the setting of K bandit arms, we will again assume that at round n each arm is described by statistics x_{na}. We can now introduce the Gittins index as

$$\nu_{na} = \sup_{\lambda} \{\lambda : D(x_{na}, \lambda) \geq 0\}. \tag{5.5}$$

We have already defined $D(x_{na}, \lambda)$ as the advantage of pulling arm a at least once over pulling the constant reference arm. As a result, by taking the supremum in (5.5) we can see that ν_{na} represents the value at which there is no advantage, i.e. this is exactly the *value* of pulling arm a. As a result, so long as we can compute this value for every arm a, we can then compare the index of each arm and select the arm with the highest index. By doing so we are essentially selecting the arm which has the highest future value under expectation. Note, however, that here we have only given an intuition as to why this value serves as the optimal index strategy. To more formally prove this intuition, see for example the work of [Frostig and Weiss, 1999].

Given a particular value of λ, Algorithm 3 details the process necessary to approximate the difference quantity introduced in (5.4) using a finite time horizon N. We can then identify the value of the index quantity by performing a bisecting search for the unique root of this function in λ. Alternatively

one can use a grid of values for λ and interpolate between points on this grid. As mentioned earlier, the method presented in this section is known as the calibration method, an analogy to the fact that the use of a standard bandit process given by λ is used to calibrate the value of each arm. This process was proposed in the original paper of [Gittins, 1979]; see also the work of [Niño-Mora, 2011]. We can now consider the complexity of this procedure. For general state-spaces the expectation in Step 4 of Algorithm 3 involves a sum over all states $x \in \mathcal{X}$, meaning that the inner loop is quadratic in the number of states and as a result the entire procedure is $O(NM^2)$ for M such states. The actual complexity for computing the index then relies on how many calls to this procedure are necessary either using a bisecting search or a grid of values.

In order to ground this discussion we can now turn to the running example of the Beta-Bernoulli model introduced earlier. For this model we can associate each state with the number of successes that have occurred. For a finite horizon N, the number of states is thus given exactly by this horizon, i.e. $M = N$. Note also that for α successes, the number of failures is given by $\beta = N - \alpha$. However, we can see that the transition model is more constrained in this setting: given α successes we can only transition in the next step to $\alpha + 1$ successes or remain at α. This results in a constant complexity for the expectation of Step 4, and as a result the complexity of Algorithm 3 reduces to $O(N^2)$. This can, however, prove to be prohibitively expensive for long time-horizons. Alternatively, we can see that the index itself only depends on the number successes seen so far in the model, so we can consider instead pre-computing these indices as a function of this quantity. However, this is only trading computation for memory, leaving us with $O(N^2)$ memory costs.

5.2 Alternative index policies and approximation guarantees

Given a Bayesian model for each arm a, we introduced in the previous section a strategy for optimally selecting arms based on their future expected

Algorithm 3
Computation of the value of a one-armed bandit with opportunity costs λ using dynamic programming. This computes the quantity $D(x_1, \lambda)$ for initial statistics x_1 and approximated using a time horizon N.

Require: x_1, N, λ, γ, and f
$V_N(x) \leftarrow 0$.
for $n = N - 1, \ldots, 1$ **do**
  for all $x \in \mathcal{X}$ reachable from x_1 in n steps **do**
    $v \leftarrow \mathbb{E}\big[R - \lambda + \gamma V_{n+1}\big(f(x, R)\big)\big]$
    $V_n(x) \leftarrow \max(0, v)$
  end for
end for
return $V_1(x_1)$.

rewards. As we have noted in the previous section, however, this optimality can come at a high price with respect to the computational cost. Alternatively, we can consider the task of finding a strategy with minimal regret, i.e. the minimal difference between the rewards obtained by some strategy and an oracle strategy that always pulls the optimal arm. In this section we will introduce this concept, as well as a number of alternative index policies that attempt to minimize the cumulative regret. We will also note how this relates to the optimal Bayesian policy.

We will first let μ_a denote the expected immediate reward of pulling arm a, i.e.

$$\mu_a = \mathbb{E}[R^{(a)}] = \int r g_a(r)\, dr,$$

and let $\mu^* = \max_a \mu_a$ denote the mean reward associated with the optimal arm. Note that in the bandit setting these quantities are unknown to the decision maker, and rely only on the unknown reward distributions. After following some arm selection strategy for n rounds, we will also consider the random variable T_{na} denoting the number of times that action a has been played up to time n. This random variable depends on the actions taken up to time n and as a result indirectly depends on the rewards—i.e. it depends exactly on the data $\mathcal{D}_n$. We can now introduce the *cumulative regret* for some strategy as the difference between gains made by following

that strategy and the gains of the single best arm

$$L_n = \max_{a \in \mathcal{A}} \mathbb{E}\left[\sum_{t=1}^{n} R_n^{(a)} - R_n\right] = n \max_a \mathbb{E}[R^{(a)}] - \sum_{a \in \mathcal{A}} \sum_{t=1}^{n} \mathbb{E}\left[\mathbb{I}_a(A_t) R_t^{(A_t)}\right]$$
$$= n\mu^* - \sum_{a \in \mathcal{A}} \mathbb{E}[T_{na}]\, \mu_a. \tag{5.6}$$

Note that the expectations written above depend on the distribution of the data $\mathcal{D}_n$, which depends not only on the rewards for each arm, but also on the strategy used in order to select each arm. By introducing an indicator over actions a we have written in the last equality the regret in terms of the means of each selected arm and the number of times T_{na} that arm has been selected.

The problem of minimizing regret was first considered in the classic work of [Lai and Robbins, 1985]. In their work, the authors introduce action selection strategies under which the expected number of actions taken from a suboptimal arm $a \in \bar{\mathcal{A}}$ can be written as

$$\mathbb{E}[T_{na}] \leq \left(\frac{1}{D_{\mathrm{KL}}(g_a, g^*)} + o(1)\right) \log n, \tag{5.7}$$

where $D_{\mathrm{KL}}(\dots)$ denotes the Kullback-Leibler distance. Also, recall that g_a is the true, unknown distribution of the ath arm, and here we will let g_* denote the distribution of the optimal arm. This bound on the number of times the ath action is played means that the optimal arm will be pulled exponentially more often than any other arm as n goes to infinity. Consequently, letting $\Delta_a = \mu^* - \mu_a$ denote the difference in expected reward between the ath arm and the optimal arm we can write the cumulative regret of this strategy as

$$L_n \leq \left[\sum_{a \in \bar{\mathcal{A}}} \left(\frac{1}{D_{\mathrm{KL}}(g_a, g^*)} + o(1)\right) \Delta_a\right] \log n. \tag{5.8}$$

We can see that this regret is of order $O(\log n)$ with average regret L_n/n approaching zero as the decision maker makes more decisions. Lai and Robbins also showed that regret of this order is optimal, i.e. no strategy

is able to improve upon this logarithmic regret. Interestingly, [Lai, 1987] was further able to show that strategies which obtain the optimal $O(\log n)$ bound are asymptotically optimal both in minimizing frequentist as well as the Bayesian risk—i.e. such approaches do as well as Gittins in the limit as n goes to infinity.

Briefly, the strategy of Lai and Robbins associates with each arm an *upper confidence bound* about the mean reward of each arm and this bound is then used within an index policy like that of Algorithm 2. However, the particular form of bound used by Lai and Robbins is in general hard to compute, relying on the entire sequence of rewards observed thus far. Instead, in the sections that follow we will present related approaches that also obtain the optimal regret bound, but are much easier to compute. Figure 5.1 shows an example of the different index strategies that we will introduce in the following subsections.

5.2.1 UCB

We will now describe the *upper confidence bound* strategy UCB1 introduced by [Auer et al., 2002]. Whereas the Gittins strategy was based on the posterior distribution, the strategy of Auer et al. is instead based on the empirical mean and high probability bounds on the deviations from this mean. Here we will let $\hat{\mu}_{na}$ denote the mean of arm a at round n and again let T_{na} denote the number of times this arm has been pulled. If no arms have been pulled yet, we will assume the mean has been arbitrarily initialized to zero. We can then introduce the following index

$$\nu_{na} = \begin{cases} \infty & \text{if } T_{na} = 0, \\ \hat{\mu}_{na} + \sqrt{\dfrac{2 \log n}{T_{na}}} & \text{otherwise.} \end{cases} \tag{5.9}$$

We can easily see that this process will begin by selecting each arm once. For every subsequent step this index is based on a Chernoff- Hoeffding bound over the deviations of the sample mean $\hat{\mu}_{na}$ from the true mean μ_a. For reward distributions g_a with support in $[0, 1]$, Auer et al. also showed that

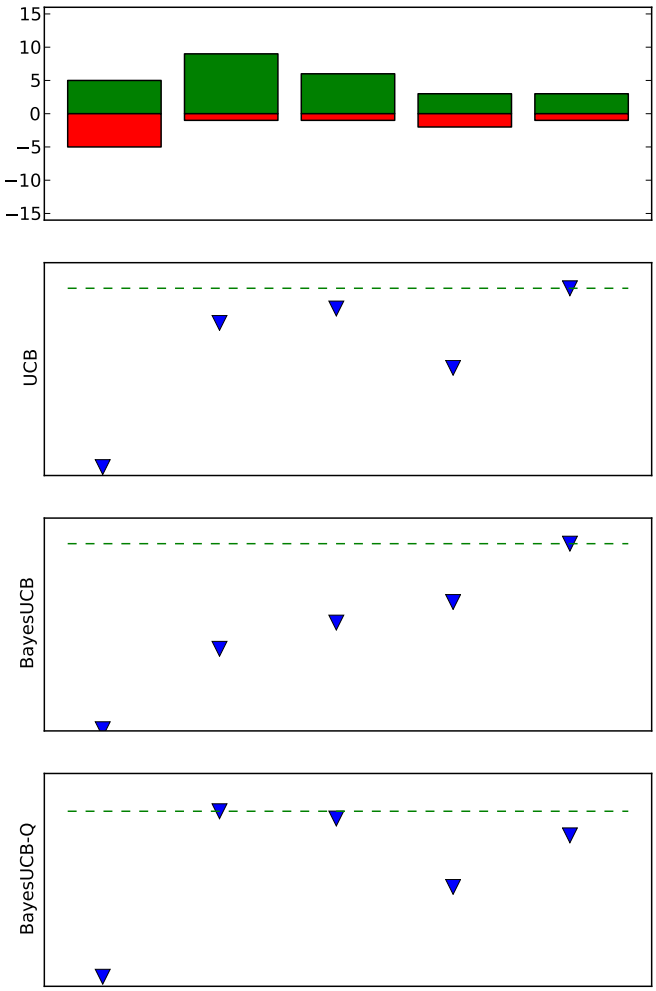

Figure 5.1: Comparison of various bandit selection indices on a five-armed problem. The top plot shows a number of successes (green) and failures (red) drawn from a Bernoulli reward model, and the bottom three plots show the relevant indices described in this section. Although the indices have somewhat similar behavior, their slight differences account for the differences in performance that we will see later. Note, we do not show an example of Thompson sampling as the randomness in this procedure makes a single sample not necessarily informative.

the UCB1 strategy exhibits cumulative regret

$$L_n \leq \left(8 \sum_{a \in \bar{\mathcal{A}}} \Delta_a^{-1}\right) \log n + (1 + \pi^2/3) \sum_{a \in \bar{\mathcal{A}}} \Delta_a. \tag{5.10}$$

As a result this strategy attains the optimal asymptotic rate, albeit with a slightly worse multiplicative constant.

5.2.2 Bayesian Quantile-based UCB

We will now return to the Bayesian setting introduced earlier wherein at round n each arm has an associated posterior $p(\theta_a|\mathcal{D}_{n-1})$. In particular, we will let $\rho_{na} = \mathbb{E}[R_n^{(a)}|\theta_a]$ denote the expected rewards of pulling arm a at time n which, as a function of the random variable θ_a, is itself a random variable. Let q_{na} denote the quantile function associated with this random variable, i.e. it is the value such that $\Pr(\rho_{na} \leq q_{na}(p)) = p$. Now, given this quantile function and the current posterior associated with arm a we can introduce the following index quantity

$$\nu_{na} = q_{na}(p_n) \text{ where } p_n = 1 - \frac{1}{n(\log N)^c}. \tag{5.11}$$

Here c is some user defined quantity and N is a finite horizon. The probability under the current posterior that the true expected reward exceeds ν_{na} is of order $1/n$, however this quantity can also be used as an upper confidence bound on the true mean.

This specific strategy was introduced by [Kaufmann et al., 2012a]. In this work the authors show that for the Beta-Bernoulli model and for any $\epsilon > 0$ and $c \geq 5$, the number of draws from a suboptimal arm can be logarithmically bounded, resulting in regret

$$L_n \leq \left[\sum_{a \in \bar{\mathcal{A}}} \left(\frac{1+\epsilon}{D_{\mathrm{KL}}(g_a, g^*)} + o_{c\epsilon}(1)\right) \Delta_a\right] \log n. \tag{5.12}$$

Here the constant term depends on c and ϵ. As a result this strategy achieves the same asymptotically optimal logarithmic rate as [Lai and Robbins, 1985].

Empirically, however, the authors find that $c = 0$ produces better results, and we found this to be the case in our experiments as well.

Finally, we should note that while Kaufmann et al. call this approach "Bayesian UCB", in this work we will refer to this index as Bayesian Quantile-based UCB (or Bayes-Q) to distinguish it from the approach of the next subsection.

5.2.3 Bayesian UCB

Briefly, we can also introduce a related upper confidence approach that is a more direct Bayesian extension of the UCB index introduced in Section 5.2.1. Let $\hat{\mu}_{na}$ and $\hat{\sigma}^2_{na}$ denote the mean and variance respectively of the random variable ρ_{na} introduced in the previous section. I.e. these quantities represent the first two moments of the posterior expected reward. We can then introduce the following index quantity:

$$\nu_{na} = \hat{\mu}_{na} + \beta_n \hat{\sigma}_{na}$$

where β_n is a sequence of constants which encourage exploration. In later experiments we will use a sequence of constants β_n of order $O(\sqrt{2\log(Kn^2)})$. This approach was originally introduced by [Srinivas et al., 2010] for the task of Bayesian optimization, and as a result we will postpone further discussion of this method until Chapter 6.

5.2.4 Thompson sampling

Finally, we can consider an alternative Bayesian approach known as probability matching, which departs from the idea of upper confidence bounds and dates back to [Thompson, 1933]. The crux of this idea is to choose the arm which has the greatest probability under the posterior of being optimal. Conditioned on the data $\mathcal{D}_{n-1}$ we can write this as

$$\Pr(\mu_a = \mu^* | \mathcal{D}_{n-1}) = \int \left\{ \rho_{na} = \max_{a'} \rho_{na'} \right\} p(\rho_n | \mathcal{D}_{n-1}) \, d\theta \tag{5.13}$$

where $p(\rho_n|\mathcal{D}_{n-1})$ is the joint posterior over all arm parameters means given as a product of the individual, independent posteriors. Thompson originally considered the problem of exactly computing this quantity in order to choose between two alternative treatments. His work also introduced the idea of approximating this probability using samples from the posterior, an idea that is now more generally known as *Thompson sampling*. Based on this idea we can consider, at each iteration, taking a single sample from the posterior for each arm and taking the expected reward under these sampled parameters, i.e.

$$\theta_n^{(a)} \sim p(\cdot|\mathcal{D}_{n-1}) \tag{5.14}$$

$$\nu_{na} = \mathbb{E}[R_n^{(a)}|\theta_n^{(a)}] = \int r\, p(r|\theta_n^{(a)})\, dr. \tag{5.15}$$

We can easily see that the samples $\theta_n^{(a)}$ correspond to a Monte Carlo approximation of the posterior where the maximization step corresponds exactly to the maximization in Equation (5.13).

Given this approach, [Kaufmann et al., 2012b] show a bound on the regret of this method that, for any $\epsilon > 0$ there exists a problem-dependent bound on the regret of the form

$$L_n \leq \left[\sum_{a \in \overline{\mathcal{A}}} \left(\frac{1+\epsilon}{D_{\mathrm{KL}}(g_a, g^*)}\right)\Delta_a\right](\log n + \log(\log n)) + c_\epsilon(\mu_1, \ldots, \mu_k). \tag{5.16}$$

Here $c_\epsilon(\cdots)$ is a constant that depends on the expected reward of each arm. We can see that this bound is quite similar to the bounds shown for Bayes-UCB, even though Thompson is a stochastic procedure that is not based on upper-confidence bounds. However, similar proof techniques were employed in showing this bound.

Finally, this approach is also often much simpler, both computationally and conceptually, than previous approaches. For the Beta-Bernoulli example used throughout this section this only requires k draws from the Beta posterior for each arm, where the expected reward is given exactly by the sampled Bernoulli success probability. Compare this, alternatively, to

the use of quantiles presented in the previous subsection, which require an expensive computation of the inverse incomplete beta function.

5.3 Empirical results for cumulative regret

In previous sections we considered a number of index policies and their associated performance guarantees. In this section we will look at the empirical, non-asymptotic performance of these policies with respect to their cumulative regret. We first consider, in Figure 5.2 a two-armed Bernoulli bandit problem as considered in [Kaufmann et al., 2012a] and vary the expected reward of the two arms while maintaining a constant gap between them. For the Bayesian approaches we used an independent, uniform $[0, 1]$ prior over the success probability θ_a for each arm. We also used a time horizon of 500 and repeated these experiments 1000 times. Each of the plots in Figure 5.2 shows a different setting of the expected rewards of each arm and plots the cumulative regret of each policy versus iterations.

From this set of experiments we can see that when the probability of observing rewards is low, the Gittins policy performs significantly better than all other policies. As rewards become more likely, however, both Thompson and the Bayesian UCB approach using quantiles approach the performance of Gittins. This makes sense: when rewards are more rare it is increasingly more beneficial to plan further into the future. In contrast, we can also see that the Bayesian UCB approach based on the variance term performs better when the rewards are rare.

We are also interested in problems where the number of arms is greater than just two. Figure 5.3 shows an experiment consisting of a 20-armed bandit with arms whose expected rewards are linearly spaced in the range $[0.1, 0.8]$. Here we consider a time-horizon of 10000 in order to test the longer range effects of each policy, and we repeat each run 500 times. Here we see that the quantile version of Bayesian UCB and Thompson sampling have approximately the same performance. Both are very much out-performed by Gittins, however we only show the first 1000 iterations of Gittins due to its overwhelming computational costs with long horizons. We also omit error

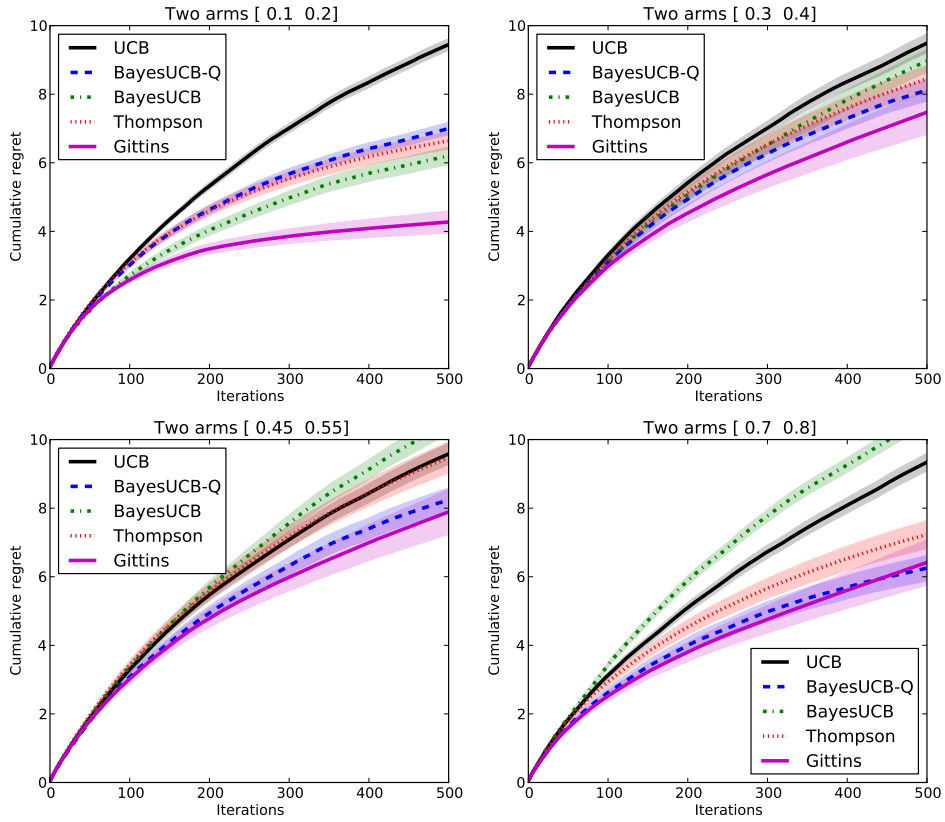

Figure 5.2: Cumulative regret of various index policies for a two-armed bandit with different reward distributions. Lower is better.

bars for this plot as they proved to be quite small and added no additional information.

5.4 Simple regret and pure exploration

In previous sections we considered the problem of choosing arms so as to maximize the expected long-run gains made by our strategy. This is equivalent to minimizing the expected regret the strategy accumulates with respect to the unknown best arm. A result of this objective is that although the

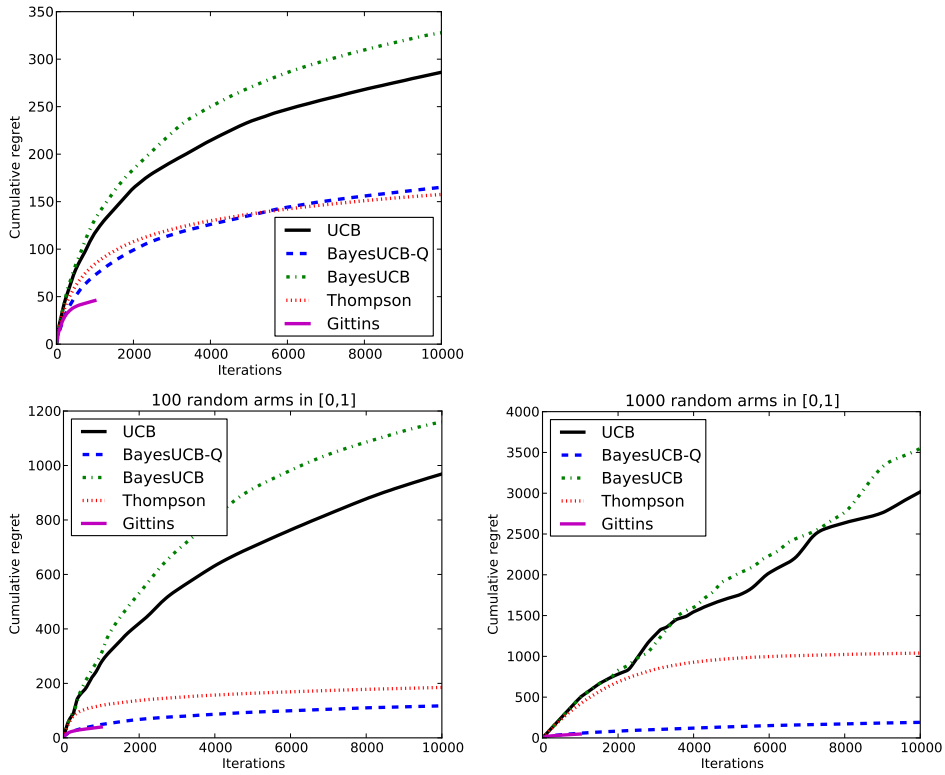

Figure 5.3: Top: cumulative regret of various index policies for a 20-armed bandit with rewards linearly spaced in the range $[.1, .8]$. Bottom: cumulative regret of various index policies for a 100- and 1000-armed bandit with random expected rewards uniformly sampled in $[0, 1]$. For all plots, lower is better.

forecaster must explore different arms in an attempt to find the optimum, it should not explore *too often* as that could otherwise lead to high cumulative regret. While this objective is a good fit for many problems, we are often instead faced with the problem of simply finding the best arm amongst a set of candidates.

In this section we will assume an alternate setting which [Bubeck et al., 2009] refers to as the problem of *pure exploration*:

For each round $n = 1, 2, \ldots$

1. the decision maker chooses an action $a_n \in \mathcal{A}$;
2. the decision maker **observes** reward r_n;

after some round N the bandit process ends

3. the decision maker makes a recommendation $i_N \in \mathcal{A}$;
4. the decision maker **receives** reward r_{N+1}.

In this setting the forecaster takes actions a_n and observes rewards r_n for N rounds following some strategy as in Algorithm 2. Along with actions a_n drawn from some exploratory strategy the forecaster must also make recommendations i_n following a possibly different strategy. After this training phase the forecaster will then enter a test phase wherein the quality of the forecaster is judged *only* on the expected reward of the last recommendation. Following from the previous section we will introduce the regret of this recommendation, i.e.

$$S_n = \mathbb{E}[\mu^* - \mu_{I_n}]. \tag{5.17}$$

This expectation is with respect to the distribution of the data $\mathcal{D}_n$ which depends on the action selection strategy, as well as the recommendation strategy which gives rise to i_n. Following [Bubeck et al., 2009] we will call this quantity the *simple regret*.

5.4.1 Best arm identification

The problem of minimizing the simple regret differs from that of more traditional bandit approaches by explicitly dividing the exploration and exploitation phases. Strategies which seek to minimize the cumulative regret such as those in Section 5.2 can instead be thought of as as combining these two phases. This is ultimately the reason why an optimal allocation strategy must limit the number of pulls of suboptimal arms to be logarithmic. If the strategy were to increase the number of pulls it could make to each arm it might better be able to explore the arm distribution, however this would in turn require more pulls of suboptimal arms and harm the forecaster in

terms of its cumulative regret.

The ideas of cumulative and simple regret are quite closely bound however. We can easily see that by using the same strategy for exploration and exploitation the cumulative regret provides an upper bound on the simple regret, i.e. $S_n \leq \frac{1}{n} L_n$. Surprisingly, however, in [Bubeck et al., 2009] the authors show that algorithms with at most logarithmic cumulative regret have at least polynomial simple regret, i.e. they are bounded by a term of order $O(n^{-\beta})$ for some $\beta > 0$. Following on this result, in [Audibert and Bubeck, 2010] the authors propose a modification of the UCB1 index of the form

$$\nu_{na} = \begin{cases} \infty & \text{if } T_{na} = 0, \\ \hat{\mu}_{na} + \sqrt{\dfrac{A_n}{T_{na}}} & \text{otherwise.} \end{cases} \tag{5.18}$$

This approach, which the authors dub UCBE, reduces to UCB1 when A_n is of order $\log n$. Letting $\Delta_{(i)}$ be the distance between the ith best arm to the best arm the authors instead propose a constant quantity $A_n = A = \frac{25}{36} \frac{N-K}{H_1}$ where $H_1 = \sum_i \Delta_{(i)}^2$ is a measure of the "hardness" of the bandit problem and N is a finite horizon. This approach encourages the algorithm to explore more, and where this exploration is tempered by the hardness of the problem.

Let E_n be the probability of recommending a suboptimal arm $E_n = \Pr(\mu_{I_n} < \mu^*)$ one can bound the simple regret by

$$\Delta_{(2)} E_n \leq S_n \leq \Delta_{(k)} E_n.$$

As a result, bounding the probability of error E_n provides a bound on the simple regret S_n. Using the UCBE strategy, Audibert and Bubeck are then able to show the following bound

$$E_n \leq 2nK \exp\left(-\frac{2A}{25}\right) \tag{5.19}$$

which holds with high probability, and as a result bounds the simple regret of the strategy.

5.4.2 Racing

The UCBE approach introduced in the previous section is, as with the approaches of Section 5.2 based on selecting arms with high upper confidence bounds. The idea of *racing* algorithms instead maintains a set of active arms and divides the process of arm selection into "races". A race consists of some number of rounds during which each active arm is pulled once. At the end of a race, arms that have been judged suboptimal according to some performance bound are then eliminated from the active set.

We will consider now a collection of arms and will introduce indicators A_{na} and S_{na} denoting respectively whether arm a is active and whether it has been selected yet in the current race. We can then introduce the following simple index

$$\nu_{na} = A_{na} S_{na}. \tag{5.20}$$

After each round, as long as there remain arms left in the race, i.e. $\sum_a S_{na} > 1$, we need not update the active arms $A_{n+1,a}$ and can set $S_{n+1,a} = 0$. Otherwise we must compute the active arms A_{n+1} and start the next race, i.e. by setting $S_{n+1} = A_{n+1}$. The exact computation of A_{n+1} will depend on the performance bound used. Note also that the index used above is not independently computed for each arm: this is to allow racing strategies to compare arms when deciding which to eliminate.

With a description of this general strategy in place we can now turn to the problem of deciding when to eliminate arms. [Maron and Moore, 1994], for example, proposed a strategy *Hoeffding races* based on the same Chernoff-Hoeffding bound used in Section 5.2.1. We can then note that for rewards in $[0, b]$, with probability $1 - \delta$ we have a bound B_{na} on deviations from the true mean,

$$|\mu_a - \hat{\mu}_{na}| \leq B_{na} = \sqrt{\frac{b^2 \log(2/\delta)}{2T_{na}}}.$$

This bound can be directly translated into an elimination strategy for racing. Let $a_n^* = \arg\max_a \hat{\mu}_{na}$ be the currently estimated best arm, then at the end

of a race we can update the active set as

$$A_{na} = \begin{cases} 1 & \text{if } \hat{\mu}_{na} + B_{na} \geq \hat{\mu}_{na^*} - B_{na^*}, \\ 0 & \text{otherwise.} \end{cases}$$

I.e. we are eliminating those arms whose upper bound is less than the lower bound of the best arm. Similar approaches can be taken based on other bounds, [e.g., Mnih et al., 2008]. Finally, although the Hoeffding-based strategy introduced above relies upon frequentist statistics, we could also consider a Bayesian version of these racing strategies involving posterior bounds on the reward.

5.4.3 A Bayesian approach

We can now return to the Bayesian setting and in particular introduce a policy which is in some sense a modification of Gittins' index made in order to attack the problem of simple regret. The modification that we are about to introduce was proposed by [Hay et al., 2012], however we will see that it is a relatively simple modification of the value function discussed earlier when we introduced Gittins.

We will first return to the MDP formulation of Section 5.1 and modify the actions to include a "stopping" action such that only upon taking this action do we receive rewards given by the expected return of the best arm. Given a set of arms $\mathcal{A}$ and sufficient statistics $\mathcal{X}$ for each of k arms we can summarize these changes as follows:

- an action space $\mathcal{A} \cup \{\perp\}$ where following the notation of Hay et al. we use $\perp$ to indicate the stopping action;
- a state space $\mathcal{X}^k$ consisting of the sufficient statistics for each arm;
- deterministic rewards given by

$$r(x_n, a_n) = \begin{cases} \max_{a \in \mathcal{A}} \mathbb{E}[R|x_n, a] & \text{if } a_n = \perp \\ -c & \text{otherwise} \end{cases}$$

where c indicates a cost for taking a non-stopping action; if a stopping action is taken the reward is given by the the best expected reward under the posterior statistics x_{na};

Note that the primary change here is the use of the stopping action in the reward function. This allows us to plan into the future, keeping in mind however that ultimately we're only interested in obtaining a single arm recommendation. Once we stop, this recommendation will be given by the arm with highest posterior mean. We should also return to a point we made earlier for the Gittins approach: this MDP is used only to plan actions into the future in order to select the best action to take *at the current time step*. In other words, when applying this approach to the problem of online planning we will employ an index quantity ν_{na} such that this MDP will be used to calculate the future expected rewards of action a.

We can now introduce the index quantity that we will consider: namely for each arm a we will compute the value of a one-armed bandit with reference arm given by the expected reward of the *best alternative arm*. In this section we will consider a finite-horizon problem, in which case we can write this quantity as

$$\nu_{na} = V_{N-n}(x_{na}, \arg\max_{a' \neq a} \hat{\mu}_{na'}), \tag{5.21}$$

where V is computed in the same way as in Section 5.1. Intuitively this means that for each arm a we are considering the value of gaining more information about that arm and possibly stopping at a later time step, or instead stopping and taking the best alternative arm. This also means that this has the same complexity as a single call to the difference quantity used to compute Gittins, i.e. $O(N^2)$. In [Hay et al., 2012] the authors refer to this as a *blinkered* policy due to the way that the index quantity compares against the best alternative arm but otherwise only considers whether to pull the ath arm or to stop.

5.5 Empirical results for simple regret

In the previous section we introduced the problem of *pure exploration* as well as the simple regret measure. In this section we will look at the empirical, non-asymptotic performance of these policies with respect to their simple regret. We will also include some of the strategies designed for cumulative regret in order to ascertain their performance—even though these strategies provably do not explore enough to obtain exponentially small probability of error. Following on the previous experiments section, we will continue testing with Bernoulli bandits, and for the Bayesian approaches we will use independent, uniform $[0, 1]$ prior over the success probability θ_a of each arm. We also used the true, empirical hardness H_1 of each problem in defining the exploration strategy for UCB-E. While this would not generally be known in practice, we wanted to see the optimal performance of this algorithm; as well we also found it to be relatively robust to this parameter. Finally, we repeat each experiment 1000 times to compute the expected regret curves.

We first consider, in Figure 5.4 the simple regret of a 20 armed bandit with means linearly spaced in $[0.1, 0.9]$. The leftmost of these two plots displays the regret over 1000 iterations, and the rightmost displays the regret over 10000 iterations. This was done to accommodate the Gittins and Blinkered strategies. For the first plot, we see that with this time-horizon, all of the strategies perform comparably. As we will see going forward, the Hoeffding races strategy performs worst. We also note the relatively poor performance of the Blinkered strategy, however this may be due to a poor tuning of the c parameter, and due to the the relatively slow speed of this algorithm this parameter was difficult to tune properly. Also of note is the performance of BayesQ, which as we will continue to see in this section does surprisingly considering it is not tuned to the problem of pure exploration. When moving on to the longer time-horizon, we see that UCB-E performs very well, although surprisingly both UCB and BayesUCB do as well. Finally we note that we have not included error bars for the plots in this section. This is because we are mostly concerned with the performance of the methods in expectation, and the error bars did not provide much ad-

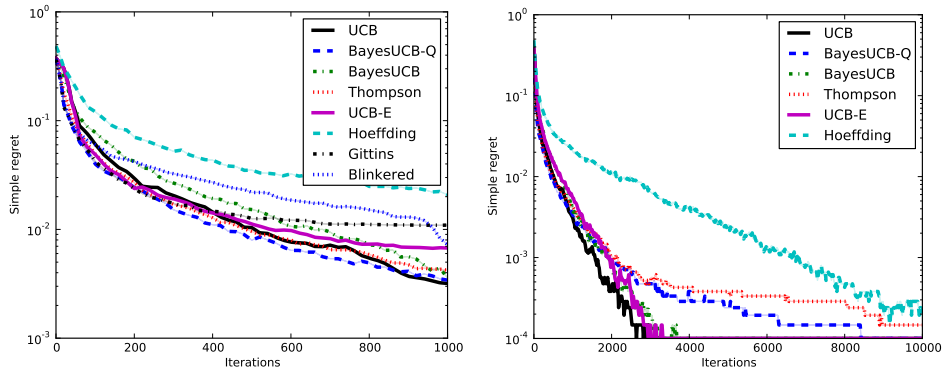

Figure 5.4: Simple regret for various bandit strategies with 20 arms, linearly spaced in $[0.1, 0.9]$. The left plot restricts the time horizon to 1000 in order to show the behavior of Gittins and Blinkered. The right plot extends this horizon to 10000. For both plots, lower is better.

ditional information aside from obscuring the performance of the different methods.

We next consider the same 20 arms, with means linearly spaced in $[.1, .9]$, except now we augment these arms with 100 additional arms. The results of these experiments are shown in Figure 5.5. The first of these plots uses 100 additional arms with mean .1, the second 100 arms with mean .5, and the last 100 arms with mean .7. With an *optimal* arm of mean .9, this is in order to the performance of the algorithms with a large number of a suboptimal arms, for varying degrees of suboptimality. We can see that with very suboptimal arms (the first plot), the results of the previous experiment are essentially unchanged. We can see in the second plot, however, that by increasing the difficulty that BayesQ and Thompson sampling become increasingly attractive options, although they are ultimately beat by UCB-E. By increasing the difficulty more we see the same trend continuing, i.e. BayesQ performs quite well initially, but it is ultimately beaten by UCB-E.

Finally, in Figure 5.6 we show the performance of these methods using 200 arms, with means linearly spaced in $[0.1, 0.9]$. This experiment has many similarly performing arms, both in terms "good" and "bad" arms.

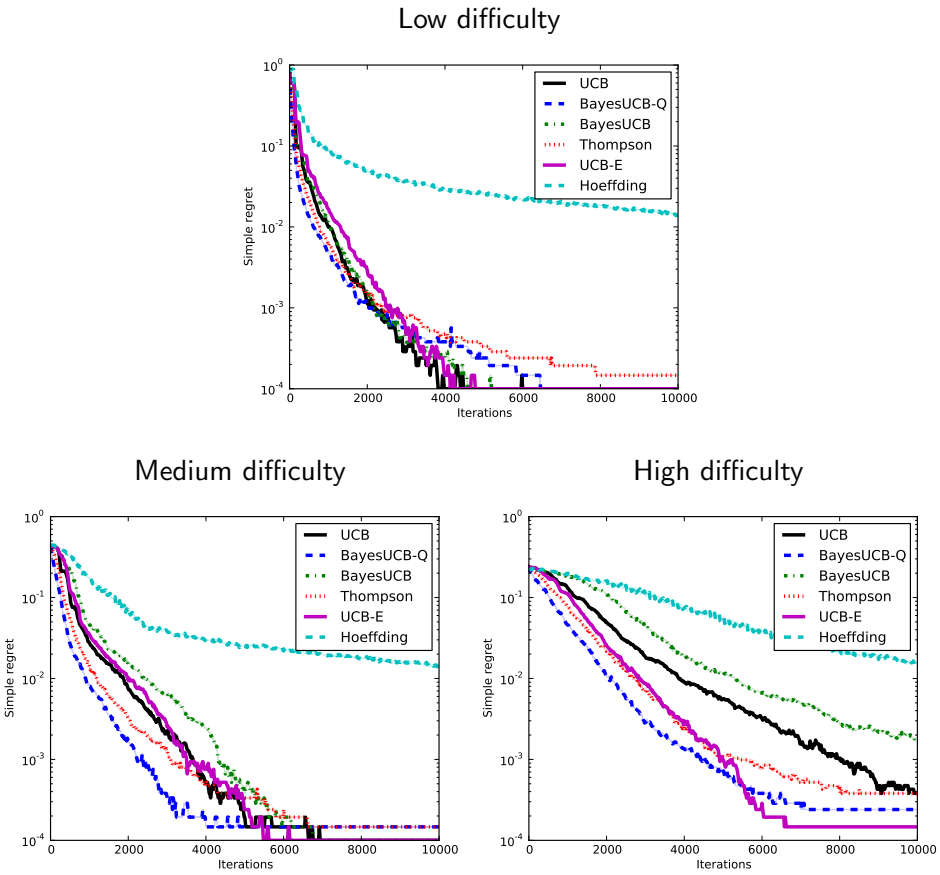

Figure 5.5: Simple regret for various bandit strategies with 120 arms of varying degrees of difficulty. All three experiments include 20 arms, linearly spaced in $[.1, .9]$. The first plot includes an additional 100 arms with mean .1, the second 100 arms with mean .5, and the last plot includes 100 additional arms with mean .7. For all plots, lower is better

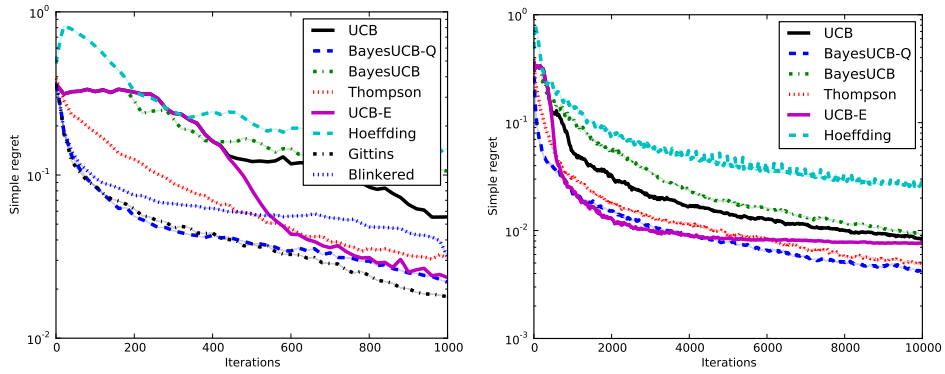

Figure 5.6: Simple regret for various bandit strategies with 200 arms, linearly spaced in $[0.1, 0.9]$. The left plot restricts the time horizon to 1000 in order to show the behavior of Gittins and Blinkered. The right plot extends this horizon to 10000. For both plots, lower is better.

We first (the leftmost plot) consider a time horizon of 1000 in order to consider Gittins and Blinkered. Here, surprisingly we note the excellent performance of Gittins. This is, however, not entirely shocking as with this many arms and this few iterations the strategies tuned for cumulative regret are also quite exploratory. However, even when we extend this to 10000 iterations (the rightmost plot) we see that BayesQ and Thompson perform very well, even beating UCB-E. This may very well be due to the fact that with so many arms and only this many iterations, in order to perform well it is worthwhile to give up on attaining a vanishing probability of error and instead focus more on exploitation. We still see, however, that these Bayesian methods as in the previous section tend to perform very well as compared to, for example, UCB.

5.6 Chapter summary and conclusions

This chapter served as a literature review of bandit methods, but also serves as a prelude to the methods we will introduce in the next chapter. Here we first introduced the multi-armed bandit problem and provided a detailed discussion of the classical method of [Gittins, 1979] solving this problem in

the Bayesian setting. However, due to the complexity of solving for the Gittins index before each arm pull, we introduced a number of alternative index strategies from the literature and discussed their approximation guarantees. We then went on to empirically compare the performance of these methods. In this setting we saw that the bound strategies based of Bayesian methods tend to outperform strategies such as UCB based on frequentist bounds. We attribute this primarily to these methods' ability to more quickly integrate information obtained from the arm pulls, as well as the use of quantiles in BayesQ.

We then introduced the problem of pure exploration, which corresponds to splitting the problem into an exploration and exploitation phase. This framework is particularly interesting as it directly applies to the problem of discrete optimization where each of the arm pulls corresponds to a noisy sample of some parameter of interest. We will see that this directly leads into the problem of Bayesian optimization that we will turn to in Chapter 6. We then introduced a number of methods from the literature for solving this problem. In testing this we found that UCB-E performs quite well at optimizing this objective. However, based on the model we also found that the Bayesian methods for cumulative regret also work surprisingly well, particularly in situations where there are many similarly performing arms. This leads us to believe that a very interesting line of future research is in developing a Bayesian version of UCB-E, similar to BayesQ.

Chapter 6

Bayesian optimization with acquisition portfolios

Bayesian optimization is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate. It is applicable in situations where one does not have a closed-form expression for the objective function, but where one can obtain noisy evaluations of this function at sampled values. It is particularly useful when these evaluations are costly, when one does not have access to derivatives, or when the problem at hand is non-convex. In fact, as we will see in this chapter these methods are quite closely related to those of bandit approaches. Like the Bayesian bandits introduced in the previous chapter, Bayesian optimization uses the history of sampled values to compute a posterior distribution over the unknown, underlying objective function. Unlike standard bandit problems however, Bayesian optimization is typically applied to problems with correlated arms—often continuous. Finally, analogous to the index strategies of the previous section, Bayesian optimization relies on an *acquisition function* with which to select the next sample point and trade off between exploration and exploitation.

The term Bayesian optimization was coined in the seventies by [Močkus et al., 1978], but a variation on the method has also been known as Efficient Global Optimization (EGO) in the experimental design literature since the nineties [Jones et al., 1998]. Bayesian optimization techniques are

also some of the most efficient approaches in terms of the number of function evaluations required. In recent years, the machine learning community has increasingly used Bayesian optimization to optimize expensive objective functions. Examples can be found in robot gait design [Lizotte et al., 2007], online path planning [Martinez-Cantin et al., 2007, 2009], intelligent user interfaces for animation [Brochu et al., 2007, 2010a], algorithm configuration [Hutter, 2009], efficient MCMC [Rasmussen, 2003], sensor placement [Osborne, 2010, Srinivas et al., 2010], and planning [Brochu et al., 2010b]. See also the thesis of [Brochu, 2010] for a detailed overview of this area.

In this chapter we will give an overview of Bayesian optimization and make notes as to how these approaches relate to the bandit methods introduced in the previous chapter. This will include the introduction of a number of acquisition functions, which again are similar in spirit to the definition of allocation strategies for bandits. The key argument that we will make in this chapter is the choice of acquisition function is not trivial, and in fact of the various acquisition functions proposed in the literature, none work well for all classes of functions. Instead, we will show that mixing over different such strategies will tend to outperform any one acquisition strategy. In particular, we will propose a solution to this problem that is similar to the bandit strategies introduced in the previous section. Essentially we propose an acquisition strategy to select amongst acquisition strategies. In this chapter we evaluate the empirical behavior of this meta-strategy on synthetic experiments (so that we can assess the effect of dimensionality), a suite of optimization problems borrowed from the global optimization literature (some of which are repeatedly cited as being very hard) and a hard, nonlinear, 9D continuous Markov decision process with a reward that has many modes and relatively large plateaus in between. The nature of the reward function in the control problem will cause gradient methods to do much worse than the Bayesian optimization strategies. Finally, this chapter will also present a theoretical analysis of the proposed techniques.

We review Bayesian optimization and popular acquisition functions in Section 6.1. In Section 6.2, we propose the use of various hedging strategies for Bayesian optimization [Auer et al., 1998, Chaudhuri et al., 2010]. In

Section 6.3, we present experimental results using standard test functions from the literature of global optimization. The experiments show that the proposed hedging approaches outperform any of the individual acquisition functions. We also provide detailed comparisons among the hedging strategies. Finally, in Section 6.4 we present a bound on the cumulative regret which helps provide some intuition as to algorithm's performance.

6.1 Bayesian optimization

Consider now the broad task of optimizing some function f defined over a general space $\mathcal{A}$,

$$x^* = \arg\max_{x \in \mathcal{A}} f(x). \tag{6.1}$$

In particular we are interested in the task of optimizing this function when we only have noisy access to $f(x)$, instead we can only make observations of the form $y \sim g(\cdot|x)$. As in the previous chapter we will consider the task of optimizing this function sequentially, letting $\mathcal{D}_n = (x_{1:n}, y_{1:n})$ denote the data obtained after n queries to the objective function. We can then define a prior $p(f)$ in function space and conditioned on this function can write the likelihood of our data as $p(\mathcal{D}_n|f) = p(\mathcal{D}_n|f(x_1), \ldots, f(x_n))$. These two terms can be combined to obtain the posterior

$$p(f|\mathcal{D}_n) \propto p(\mathcal{D}_n|f)\, p(f). \tag{6.2}$$

The posterior captures the updated beliefs about the unknown objective function. One may also interpret this step as estimating the objective function with a *surrogate function* (also called a *response surface*).

The approach of Bayesian optimization then relies on maintaining this posterior distribution and further defining an *acquisition function* $\nu(x|\mathcal{D}_n)$ such that the next point selected is $x_{n+1} = \arg\max_{x \in \mathcal{A}} \nu(x|\mathcal{D}_n)$. Pseudo-code describing this approach is shown in Algorithm 4. We can also note that this formulation is general enough to encompass the Bayesian strategies of the previous chapter. Consider, for example, optimizing over a discrete

space $\mathcal{A} = \{x_1, \ldots, x_K\}$ and an independent prior

$$p(f) = \prod_{a=1}^{K} p(f(x_a)).$$

The function evaluations $f(x_a)$ correspond exactly to the expected rewards μ_a with posterior predictions $p(f(x_a)|\mathcal{D}_n)$ corresponding to the random variable ρ_{na} for each arm. Unlike the setting of the previous chapter, however, we will now focus on the problem of *continuous arms* and as a result we will consider correlations between arms in order to take advantage of any smoothness in the underlying function.

Throughout the rest of this chapter we will assume a d-dimensional continuous space $\mathcal{A} \subseteq \mathbb{R}^d$ with function evaluations subject to additive Gaussian noise. Other observation models are possible [e.g., Brochu et al., 2010b, Chu and Ghahramani, 2005, Diggle et al., 1998, Rue et al., 2009], but we will focus on real, Gaussian observations for ease of presentation. In particular if x_n is the nth sample point we will assume observations of the form $y_n = f(x_n) + \epsilon_n$ with independent noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$.

In order to capture the continuous nature of the underlying function we will place a Gaussian process (GP) prior on f. This prior captures the fact that for smooth objective functions, data with high variance or oscillations should be considered less likely than data remains near the mean. Here, the level of smoothness is controlled by the prior's hyperparameters. Other nonparametric priors over functions, such as random forests, have been considered [Brochu et al., 2010b], but the GP strategy is the most popular alternative. We will give a brief description of this model in Section 6.1.1 and in Section 6.1.2 we will describe various acquisition functions that take advantage of this posterior.

6.1.1 Gaussian processes

A Gaussian Process (GP) is a stochastic process corresponding to a collection of random variables where any finite number of these variables has a joint Gaussian distribution. In this chapter we will use a zero-mean GP as

Algorithm 4 Bayesian Optimization

1: **for** $n = 1, 2, \ldots$ **do**
2: Select $x_n = \arg\max_{x \in \mathcal{A}} \nu(x|\mathcal{D}_{n-1})$
3: Sample the objective function $y_n \sim g(\cdot|x_n)$ (i.e. sample $f(x_n) + \epsilon_n$)
4: Augment the data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup (x_n, y_n)$
5: **end for**
6: **return** the incumbent, $x^+ = \arg\max_{\{x_n\}} \mu(x_n)$

a prior distribution over functions, written

$$f(x) \sim \mathrm{GP}(0, k(x, x'))$$

with *covariance function* k. For any collection of points $x_{1:n}$ let $f_{1:n}$ denote the random variables associated with the corresponding function evaluations and let K be a kernel matrix consisting of pairwise covariances $K_{ij} = k(x_i, x_j)$. We can then write the prior density of the function evaluations as

$$f_{1:n} \sim \mathcal{N}(0, K). \tag{6.3}$$

Finally, we should note that although throughout this chapter we will assume a zero-mean prior, this choice is purely for convenience and without loss of generality. For examples of nonzero means, see [Brochu et al., 2010a, Martinez-Cantin et al., 2007]; see also the work of [Rasmussen and Williams, 2005] for an extended description of GPs in general.

With the choice of a GP prior in place we are now left with the question of selecting the covariance function k. A very popular choice is the squared exponential kernel

$$k(x, x') = \exp\big(-\tfrac{1}{2}(x - x')^T A (x - x')\big), \tag{6.4}$$

for some symmetric matrix A. One possible choice for the "distance" matrix that we will use in this work is that of $A = \mathrm{diag}(\ell)^{-2}$ for some vector ℓ. Such a vector implements automatic relevance determination (ARD) where the size of the length scales ℓ_i determine how relevant the ith dimension is. The precise choice of these hyperparameters will be discussed in the experimental

section, but we note that it is not trivial in Bayesian optimization because of the paucity of data. For an in depth analysis of this issue we refer the reader to [Brochu et al., 2010a, Osborne, 2010].

Now, assuming that we have obtained observations $\mathcal{D}_n = (x_{1:n}, y_{1:n})$, we are often faced with the problem of predicting the value f_* of some arbitrary sample point x_*. Letting $y_{1:n}$ denote the observed values of the previously sampled points we can note that the vector of function values is jointly Gaussian

$$\begin{bmatrix} y_{1:n} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & k_* \\ k_*^T & k(x_*, x_*) \end{bmatrix}\right),$$

where $k_* = [k(x_1, x_*), \ldots, k(x_n, x_*)]^T$ is the vector of covariances between $x_{1:n}$ and x_* and again K is the matrix of cross-covariances with $K_{ij} = k(x_i, x_j)$. Note also the addition of the noise term $\sigma^2 I$, due to the fact that the observations have both covariance via to the underlying function evaluations and are also observed with additive Gaussian noise. The variance term corresponding to the sample point x_*, however, does not include this noise term because we are trying to predict f_*, not the corresponding observation y_*. Now, using the Sherman-Morrison-Woodbury formula, [see Rasmussen and Williams, 2005, for a comprehensive treatment] one can easily arrive at an expression for the predictive distribution

$$\Pr(f_*|\mathcal{D}_n, x_*) = \mathcal{N}(f_*|\mu_n(x_*), \sigma_n^2(x_*))$$

where the predictive mean and variance are given by

$$\begin{aligned} \mu_n(x_*) &= k_*^T [K + \sigma^2 I]^{-1} y_{1:n}, \\ \sigma_n^2(x_*) &= k(x_*, x_*) - k_*^T [K + \sigma^2 I]^{-1} k_*. \end{aligned}$$

In this sequential decision making setting, the number of query points is relatively small and, consequently, the GP predictions are easy to compute.

6.1.2 Acquisition functions

The role of the acquisition function is to guide the search for the optimum. Typically, acquisition functions are defined such that high values correspond to *potentially* high values of the objective function, whether because the prediction is high, the uncertainty is great, or both. Again, as we have noted previously this quantity has an analogous role to that of the index quantities introduced in the previous chapter. Just like the allocation indices, the acquisition function is maximized at every iteration in order to select the next point at which to evaluate the objective function, i.e. $x_{n+1} = \arg\max \nu(x|\mathcal{D}_n)$. However, unlike the previous chapter although this quantity is known and easy to evaluate, it is generally a continuous function of x. Rather than utilizing a direct discrete maximization, optimization of this term can instead be easily carried out with standard numerical techniques. Common approaches include sequential quadratic programming or LBFGS [see Nocedal and Wright, 1999] or global optimization methods such as the DIRECT algorithm of [Jones et al., 1993]. Finally, we note that the acquisition function is sometimes called the *infill* or simply the "utility" function. In the following sections, we will look at the three most popular choices. Figure 6.1 shows how these give rise to distinct sampling behavior.

Probability of improvement

Let $x_n^+ = \arg\max_{\{x_i\}} \mu_n(x_i)$ denote the current incumbent, i.e. the previously sampled point with the highest posterior mean. Early work of [Kushner, 1964] suggested using as acquisition function the *probability of improvement* (PI) over this incumbent. Strictly using this rule would, however, be biased towards exploitation as the algorithm is more likely to continually explore small regions around the current incumbent. To remedy this, Kushner recommends using instead improvement over the incumbent plus some trade-off parameter $\xi \geq 0$. We can then write the acquisition function as

$$\nu_{\text{PI}}(x|\mathcal{D}_n) = \Pr(f(x) \geq \mu_n(x_n^+) + \xi)$$

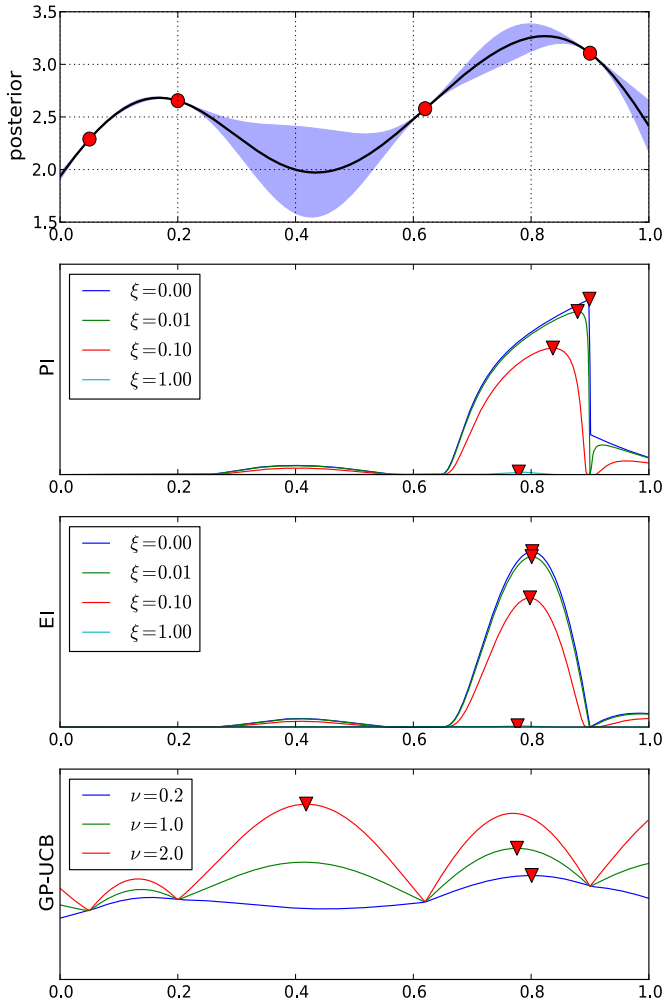

Figure 6.1: Acquisition functions with different values of the exploration parameter ξ. The GP posterior is shown at the top. The other images show the acquisition functions for that GP. From the top: Probability of improvement, expected improvement and GP-UCB. The maximum of each acquisition function, where the GP is to be sampled next, is shown with a triangle marker. Note the increased preference for exploration exhibited by GP-UCB.

which under the GP posterior, where Φ is the cumulative distribution function (CDF) of a standard Normal, reduces to

$$= 1 - \Phi\left(\frac{\mu_n(x_n^+) + \xi - \mu_n(x)}{\sigma_n(x)}\right) \equiv \frac{\mu_n(x) - \mu_n(x_n^+) - \xi}{\sigma_n(x)}. \quad (6.5)$$

Here we have also noted that since the CDF is a monotonic increasing transformation, the left and right equations in (6.5) have equivalent maxima[1]. The exact choice of ξ is left to the user. Kushner anecdotally recommends using a schedule for ξ which should start high in order to drive exploration and decrease towards zero as the algorithm progresses. Empirical studies of [Lizotte, 2008], however, found that using such a schedule did not offer improvement over a constant value of ξ on a suite of test functions.

Expected improvement

More recent work has tended to take into account not only the probability of improvement, but the magnitude of improvement a point can potentially yield. As an example of this approach, [Močkus et al., 1978] proposed maximizing the *expected improvement* (EI) with respect to the current incumbent. For our Gaussian process posterior, one can easily evaluate this expectation, [see Jones, 2001], yielding:

$$\begin{aligned}\nu_{\text{EI}}(x|\mathcal{D}_n) &= \mathbb{E}[f(x) - f(x_n^+)|\mathcal{D}_n] \\ &= \begin{cases} a\Phi(a\sigma_n^{-1}(x)) + \sigma_n(x)\,\phi(a\sigma_n^{-1}(x)) & \text{if } \sigma_n(x) > 0, \\ 0 & \text{if } \sigma_n(x) = 0 \end{cases}\end{aligned}$$

where $a = \mu_n(x) - \mu_n(x_n^+) - \xi$ and where ϕ and Φ denote the density and cumulative density of the standard Normal distribution respectively. Again, ξ is an optional trade-off parameter analogous to the one defined above.

[1]Note, that in the thesis of [Lizotte, 2008] there is a minor error in reporting the form of PI, wherein the quantity inside the evaluation of the Normal CDF is multiplied by -1.

Upper confidence bounds

As we saw in the previous chapter, upper confidence bounds have a long history of being used for online optimization, dating back to the work of [Lai and Robbins, 1985]. In a similar vein in the Bayesian optimization literature, [Cox and John, 1997] introduced an algorithm they call "Sequential Design for Optimization" (SDO). Given a random function model, SDO selects points for evaluation based on a confidence bound consisting of the mean and weighted variance: $\mu(x) + \kappa\sigma(x)$, which we can see also bears striking similarities to the UCB1 method of [Auer et al., 2002]. As with the other acquisition models, however, the parameter κ is left to the user.

More recently, a principled approach to selecting this parameter is proposed by [Srinivas et al., 2010] drawing directly on the earlier work of Auer et al. In this work, the authors define the instantaneous regret of the selection algorithm as $f(x^*) - f(x)$, i.e. a continuous generalization of the instantaneous regret defined in the previous chapter. They then attempt to select a sequence of weights κ_t so as to minimize the cumulative regret. Using a sequence of weights β_t and an exploration parameter ξ, we can define the UCB acquisition function as

$$\nu_{\text{UCB}}(x|\mathcal{D}_n) = \mu_n(x) + \sqrt{\xi\beta_t}\,\sigma_n(x).$$

It can be shown, with high probability, that this method has cumulative regret bounded by $\mathcal{O}(\sqrt{N\beta_N\gamma_N})$. Here β_N is a carefully selected learning rate and γ_N is a bound on the information gained by the algorithm at selected points after N steps. Both of these terms depend upon the particular form of kernel-function used, but for most kernels their product can be shown to be sublinear in N. We refer the interested reader to the original paper of Srinivas et al. for exact bounds for a variety of kernels. Interestingly, the $\sqrt{N}$ behavior of this bound is similar to bounds for an alternative extension of bandit methods to the problem of continuous arms, [see e.g. Bubeck et al., 2011].

Finally, the sublinear bound on cumulative regret implies convergence of the method, namely that $\lim_{N\to\infty} L_N/N = 0$. This in turn provides a

bound on the convergence rate for the optimization process, since the regret at the maximum $f(x^*) - \max_n f(x_n)$ is upper bounded by the average regret

$$\frac{1}{N} L_N = f(x^*) - \frac{1}{N} \sum_{n=1}^{N} f(x_n).$$

As we will note later, however, this bound can be loose in practice. Further, as we will note later the problem of Bayesian optimization, as an optimization problem, is more concerned with minimizing the simple regret than the cumulative regret. As a result, good bounds on the cumulative regret may imply that the algorithm is in some sense not exploring enough.

6.2 Portfolio strategies

In this section we claim that there is no choice of acquisition function that can be guaranteed to perform best on an arbitrary, unknown objective. We will empirically study this claim later in this chapter, however we will first make this claim as an appeal to intuition. While studying the objective might allow an expert to make an educated guess, even this can be difficult as Bayesian optimization is normally used specifically when sampling the objective is expensive. In fact, it may be the case that no single acquisition function will perform the best over an entire optimization—a mixed strategy in which the acquisition function samples from a pool (or portfolio) at each iteration might work better than any single acquisition. This can be treated as a problem of decision making with expert advice, where each of acquisition functions corresponds to an *expert* (similar to the arms of the previous chapter). In turn, each of these experts is making recommendations with respect to an underlying infinite-armed bandit, i.e. the objective function of Bayesian optimization. In this section we propose the novel approach of solving this selection problem using three strategies from the literature.

6.2.1 Making decisions with expert advice

Let $\mathcal{K}$ be a set of M experts, which in this section will coincide with the choice of acquisition function. We can now consider the problem of making

decisions with the advice of these experts.

For each round $n = 1, 2, \ldots$

1. the environment chooses rewards $r_n^{(i)}$ for each expert;
2. the decision maker chooses to follow advice $i_n \in \mathcal{K}$;
3. the decision maker receives reward $r_n = r_n^{(i_n)}$ for the chosen expert;
4. the decision maker observes rewards $r_n^{(i)}$ for each expert.

Where the goal is to maximize the rewards r_n. Here, we can easily see that Steps 2–3 coincide exactly with the bandit problem. The key differences are first, upon choosing some expert i the decision maker receives the reward for that expert but also then observes rewards for all other experts. This setting is known as *full information* which distinguishes it from the bandit or *partial information* setting. The second key difference with the previous chapter is that the rewards are not assumed to be drawn from some unknown distribution and are instead assumed to be chosen arbitrarily. This condition is often known as the *non-stochastic* or *adversarial* setting.

As a result of these two conditions we can now note that the general problem of utilizing expert advice can be seen as one governed by two axes, one being whether or not the problem is stochastic or adversarial, with the other axis being that of full versus partial information. With this in mind we can see that the bandit strategies of the previous chapter are stochastic (non-adversarial) with partial information. This also points to why these algorithms are not applicable to the problem at hand, i.e. that of selecting amongst a set of acquisition functions. While the additive noise of ϵ_n is assumed to have some fixed distribution the recommendations of each acquisition function also consist of a non-stochastic component due to the underlying function evaluations $f(x_n)$. We will return to this problem in the next section where we explicitly apply these methods to the meta-problem of acquisition function selection.

The Hedge algorithm of [Freund and Schapire, 1995] is an algorithm which solves the problem of following expert advice with full information and adversarial rewards. See also the later work of [Auer et al., 1998]. In essence, the algorithm maintains a vector g_{ni} of the cumulative rewards for each expert i up to round n. At every iteration the algorithm then follows the advice of expert i with probability $p_{ni} \propto \exp(\eta g_{n-1,i})$ for some exploration parameter η. Note that unlike the previous chapter the action selection is probabilistic, due in part to the fact that the rewards can be adversarial.

[Auer et al., 1998] also proposed the Exp3 algorithm, a variant of Hedge that applies to the partial information (i.e. bandit) setting. Note that this is equivalent to removing Step 4 from the problem described above. Rather than abandoning the approach of Hedge entirely, Exp3 uses this algorithm as a subroutine using simulated rewards $\hat{r}_{ni}$ for each expert i. Letting $\hat{p}_{ni}$ be the distribution with which Hedge would have selected actions, upon taking action i_n and receiving reward r_n, Exp3 then updates Hedge with rewards

$$\hat{r}_{ni} = \begin{cases} r_n / \hat{p}_{ni} & \text{if } i = i_n, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, Exp3 selects actions according to a distribution that is a mixture between $\hat{p}_{ni}$ and a uniform distribution. Intuitively this ensures that the algorithm does not miss good actions because the initial rewards were low (i.e. it continues exploring). The mixing constant of this distribution is an exploration factor similar to η.

Another possible strategy is the NormalHedge algorithm of [Chaudhuri et al., 2009]. If we let $g_n = \sum_i p_{ni}\, g_{ni}$ denote the expected gain of a hedging algorithm, we can introduce let $L_{ni} = \sum_n g_{ni} - g_n$ as the cumulative regret of the ith expert. We can then easily see that Hedge elects to follow each expert with probability proportional to $\exp(\eta L_{ni})$. The strategy of NormalHedge, however, uses a similar strategy as Hedge, but instead uses probabilities of the form $p_{ni} = \exp(L_{ni}^2 / 2c_n)$ for some adaptive scale parameter c_n. The key property of this method is the selection of the scale parameter, which is performed adaptively based on the regret seen thus far.

This method, however, is built to take advantage of situations where the number of experts is large, and may not be a good match to problems where M is relatively small, i.e. the situation we find ourselves in when selecting acquisition functions.

6.2.2 Bayesian optimization with expert advice

We can now consider the problem of performing Bayesian optimization with a portfolio $\mathcal{K}$ consisting of M different acquisition functions. In order to tackle this problem, at every iteration n we will follow the recommendation of strategy $i \in \mathcal{K}$ with probability p_{ni} and adapt the approaches of the previous subsection in order to update this probability. The general approach of Bayesian optimization with portfolios is shown in Algorithm 5.

Algorithm 5
General procedure for performing Bayesian optimization using a portfolio of acquisition functions. The exact form of the algorithm will depend on the method used to update the expert selection probabilities in Step 8.

1: Initialize the gains, $g_{0i} = 0$ for each i
2: Initialize the probabilities, $p_{1i} = \frac{1}{M}$ for each i
3: **for** $n = 1, 2, \ldots$ **do**
4: Nominate points from each acquisition function:
$x_n^{(i)} = \arg\max_x \nu_i(x|\mathcal{D}_{n-1})$
5: Select nominee $x_n = x_n^{(i)}$ with probability p_{ni}
6: Sample the objective function $y_n = f(x_n) + \epsilon_n$
7: Augment the data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup (x_n, y_n)$ and update the GP
8: Update gains g_{ni} and probabilities $p_{n+1,i}$
9: **end for**

The easiest algorithm to make use of is that of Exp3. Let $x_n^{(i)} \in \mathcal{A}$ be the next point proposed by the ith acquisition function. With probability p_{ni} we will select this acquisition function, i.e. by letting $x_n = x_n^{(i)}$ and obtaining an observation $y_n = f(x_n^{(i)}) + \epsilon_n$. We can then use y_n as the observed reward for following the ith expert (analogous to pulling the ith arm) and update the gains g_{ni} and hence the probabilities $p_{n+1,i}$ that will be used in the next round. Alternatively, after updating the GP with the data pair (x_n, y_n) we

can use as reward the updated mean of the GP evaluated at the selected point, i.e. $\mu_n(x_n)$. In some sense this is a more natural reward to make use of, as the "meta" algorithm (i.e. Exp3 in this case) is selecting points based on their effect on the underlying Gaussian process.

With this in mind we can now propose a reward model that allows us to make use of the full-information approaches, Hedge and NormalHedge. Just as with Exp3, both hedging strategies maintain a probability p_{ni} with which a single expert $x_n = x_n^{(i)}$ is selected. We can again define the reward $r_n = \mu_n(x_n)$ obtained by this strategy. For the hedging strategies we can define rewards observed for all other strategies as $r_n^{(i)} = \mu_n(x_n^{(i)})$. In other words after selecting the single point x_n and making an observation y_n at this point we will update the GP and "observe" the updated mean at all other proposed points based on this new information. There is however still information gained at these unselected points due to the covariance structure of the GP prior.

We will also note that the setting of our problem is somewhere "in between" the full and partial information settings. Consider, for example, the situation that all points sampled by our algorithm are "too distant" in the sense that the kernels evaluated at these points exert negligible influence on each other. In this case, we can see that only information obtained by the sampled point is available, and as a result GP-Hedge will be over-confident in its predictions when using the full-information strategy. However, this behavior is not observed in practical situations because of smoothness properties, as well as our particular selection of acquisition functions. In the case of adversarial acquisition functions one might instead choose to use the Exp3 variant.

In practice any of the above allocation strategies could be used, however in our experiments we will find that Hedge tends to outperform the others. In later experiments we will use GP-Hedge to denote this combination of Bayesian optimization with the Hedge algorithm.

Finally, note that in contrast to the standard approach Bayesian optimization by using any of these strategies it is necessary to optimize each of the M acquisition functions at each time step rather than just 1. While this

might seem expensive, this is unlikely to be a major problem in practice for small M, as (i) Bayesian optimization is typically employed when sampling the objective is so expensive as to dominate other costs; (ii) it has been shown that fast approximate optimization of the acquisition functions ν are usually sufficient [Brochu et al., 2010b, Hutter, 2009, Lizotte, 2008]; and (iii) it is straightforward to run the optimizations in parallel on a modern multicore processor.

6.3 Experiments

To validate the use of GP-Hedge, we tested the optimization performance on a set of test functions with known maxima $f(x^*)$. To see how effective each method is at finding the global maximum, we use the "gap" metric [see Huang et al., 2006], defined as

$$G_n = \Big[f(x_n^+) - f(x_1)\Big] \Big/ \Big[f(x^*) - f(x_1)\Big],$$

where again x_n^+ is the incumbent or best sample point found up to time n. The gap G_n will therefore be a number between 0 (indicating no improvement over the initial sample) and 1 (if the incumbent is the maximum). Note, while this performance metric is evaluated on the true function values, this information is not available to the optimization methods. Finally, we should also note that this metric is also a normalized version of the simple regret of the previous chapter.

6.3.1 Standard test functions

We first tested performance using functions common to the literature on Bayesian optimization: the Branin, Hartman 3, and Hartman 6 functions. All of these are continuous, bounded, and multimodal, with 2, 3, and 6 dimensions respectively. See the work of [Brochu, 2010, Lizotte, 2008] for further information and definition of these test functions. These functions have been proposed by [Dixon and Szegö, 1978] as benchmarks for comparing global search methods and are widely used for this purpose, [Jones

et al., 1993]. For each experiment, we optimized 25 times and computed the mean and variance of the gap metric over time. In these experiments we used hyperparameters θ chosen offline so as to maximize the log marginal likelihood of a (sufficiently large) set of sample points [see Rasmussen and Williams, 2005]. Note that in the synthetic functions, for each trial, we sample a different function and optimize it using the different techniques, so each plot shows the mean result over 25 different functions, with one trial each.

We compared the standard acquisition functions using parameters suggested by previous authors, i.e. $\xi = 0.01$ for EI and PI, $\delta = 0.1$ and $\xi = 0.2$ for GP-UCB [Lizotte, 2008, Srinivas et al., 2010]. The plots for EI, PI and GP-UCB shown all use these parameters. For the GP-Hedge trials, we tested performance under using both 3 acquisition functions and 9 acquisition functions. For the 3-function variant we use the standard acquisition functions with default hyperparameters. The 9-function variant uses these same three as well as 6 additional acquisition functions consisting of: both PI and EI with $\xi = 0.1$ and $\xi = 1.0$, GP-UCB with $\xi = 0.1$ and $\xi = 1.0$. While we omit trials of these additional acquisition functions for space reasons, these values are not expected to perform as well as the defaults and our experiments confirmed this hypothesis. However, we are curious to see if adding known suboptimal acquisition functions will help or hinder GP-Hedge in practice.

Results for the gap measure G_n are shown in Figure 6.2. While the improvement GP-Hedge offers over the best single acquisition function varies, there is almost no combination of function and time step in which the 9-function GP-Hedge variant is not the best-performing method. The results suggest that the extra acquisition functions assist GP-Hedge in exploring the space in the early stages of the optimization process. Figure 6.2 also displays, for a single example run, how the the arm probabilities p_{ni} used by GP-Hedge evolve over time. We have observed that the distribution becomes more stable when the acquisition functions come to a general consensus about the best region to sample. As the optimization progresses, exploitation becomes more rewarding than exploration, resulting in more probability being assigned to methods that tend to exploit. However, note

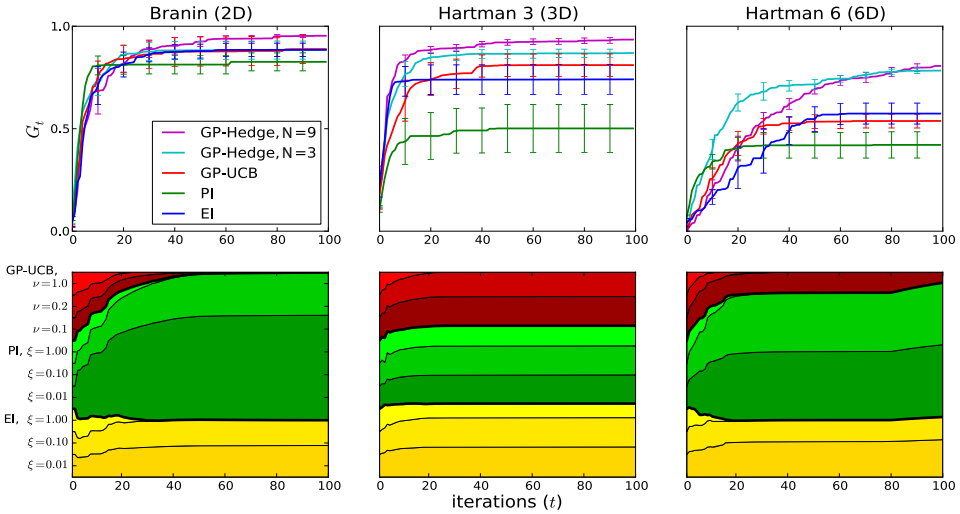

Figure 6.2: Comparison of the three base acquisition functions with GP-Hedge on three commonly used literature functions. The top plots show the mean and variance of the gap metric averaged over 25 trials. We note that the top two performing algorithms use a portfolio strategy. With $M = 3$ acquisition functions, GP-Hedge beats the best-performing acquisition function in almost all cases. With $M = 9$, we add additional instances of the three acquisition functions, but with different parameters. Despite the fact that these additional functions individually perform worse than the ones with default parameters, adding them to GP-Hedge improves performance in the long run. The bottom plots show an example evolution of GP-Hedge's portfolio with $M = 9$ for each objective function. The height of each band corresponds to the probability p_{ni} at each iteration.

that if the initial portfolio had consisted only of these more exploitative acquisition functions, the likelihood of becoming trapped at suboptimal points would have been higher.

In Figure 6.3 we compare against the other Hedging strategies introduced in Section 6.2 under both the gap measure and mean average regret. We also introduce a baseline strategy which utilizes a portfolio uniformly distributed over the same acquisition functions. The results show that mixing across multiple acquisition functions provides significant performance ben-

efits under the gap measure, and as the problems' difficulty/dimensionality increases we see that GP-Hedge outperforms other mixed strategies. The uniform strategy performs well on the easier test functions, as the individual acquisition functions are reasonable. However, for the hardest problem (Hartman 6) we see that the performance of the naive uniform strategy degrades. NormalHedge performs particularly poorly on this problem. We observed that this algorithm very quickly collapses to an exclusively exploitative portfolio which becomes very conservative in its departures from the incumbent. We again note that this strategy is intended for large values of M, which may explain this behavior.

In the case of the regret measure we see that the hedging strategies perform comparable to GP-UCB, a method that we know to have good regret bounds for this setting. We note, however, that although the average cumulative regret can prove quite useful in assessing the convergence behavior of Bayesian optimization methods, the bounds provided by this regret can be loose in practice. Further, in the setting of Bayesian optimization we are typically concerned not with the cumulative regret during optimization, but instead with the regret incurred by the incumbent after optimization is complete. This directly relates to the question of simple regret detailed in the previous chapter, and as a result good regret bounds for the cumulative regret do not necessarily imply good performance with respect to the optimization problem.

Finally, based on the performance in these experiments, we will use Hedge as the underlying algorithm for GP-Hedge in the remainder of the experiments.

6.3.2 Sampled test functions

As there is no generally-agreed-upon set of test functions for Bayesian optimization in higher dimensions, we seek to sample synthetic functions from a known GP prior, similar to the strategy of [Lizotte, 2008]. A GP prior is infinite-dimensional, so on a practical level for performing experiments we simulate this by sampling points and using the posterior mean as the

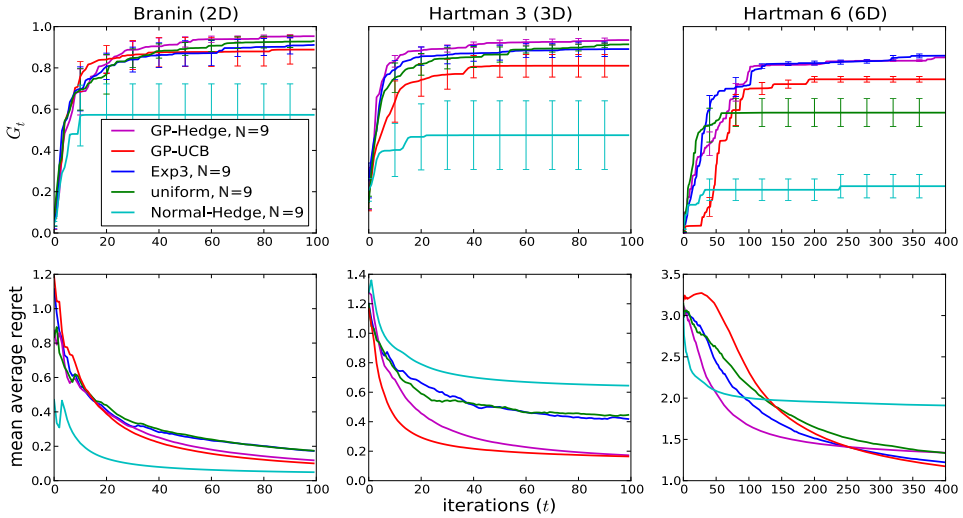

Figure 6.3: Comparison of different hedging strategies on three commonly used literature functions. The top plots show the mean and variance of the gap metric averaged over 25 trials. Note that both Hedge and Exp3 outperform the best single acquisition function, GP-UCB. The bottom plots show the mean average regret for each method (lower is better). Average regret is shown in order to compare with previous work of [Srinivas et al., 2010], however as noted in the text the gap measure provides a more direct comparison of optimization performance. We see that mixed strategies (i.e. GP-Hedge) perform comparably to GP-UCB under the regret measure and outperform this individual strategy under the gap measure. As the problems get harder, and with higher dimensionality, GP-Hedge significantly outperforms other acquisition strategies.

synthetic objective test function.

For each trial, we use an ARD kernel with θ drawn uniformly from $[0, 2]^d$. We then sample $100d$, d-dimensional points, compute the kernel matrix K and then draw $y \sim \mathcal{N}(0, K)$. The posterior mean of the resulting predictive posterior distribution $\mu(x)$ introduced in Section 6.1.1 is used as the test function. However it is possible that for particular values of θ and K, large parts of the space will be so far from the samples that they will form plateaus along the prior mean. To reduce this, we evaluate the test function at 500

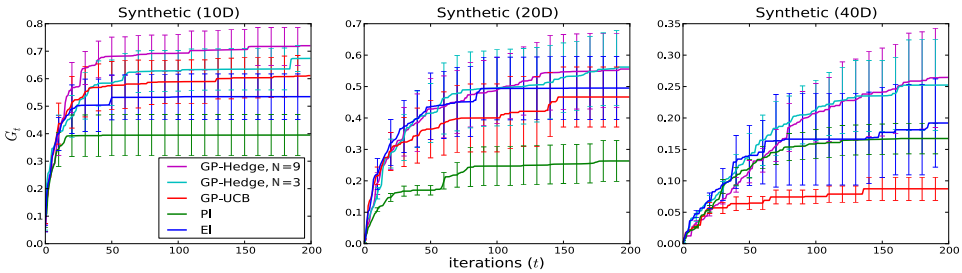

Figure 6.4: Comparison of the performance of the acquisition approaches on synthetic functions sampled from a GP prior with randomly initialized hyperparameters. Shown are the mean and variance of the gap metric over 25 sampled functions. Here, the variance is a relative measure of how well the various algorithms perform while the functions themselves are varied. While the variance is high (which is to be expected over diverse functions), we can see that GP-Hedge is at least comparable to the best acquisition functions and ultimately superior for both $M = 3$ and $M = 9$. We also note that for the 10- and 20-dimensional experiments GP-UCB performs quite well but suffers in the 40-dimensional experiment. This helps to confirm our hypothesis that a mixed strategy is particularly useful in situations where we do not possess strong prior information with regards to the choice of acquisition function.

random locations. If more than 25 of these are 0, we recompute K using $200d$ points. This process is repeated, adding $100d$ points each time until the test function passes the plateau test (this is rarely necessary in practice).

Using the response surface $\mu(x)$ as the objective function, we can then approximate the maximum using conventional global optimization techniques to get $f(x^*)$, which permits us to use the gap metric for performance. Note that these sample points are only used to construct the objective, and are not known to the optimization methods.

As can be seen in Figure 6.4, GP-Hedge with $M = 9$ is again the best-performing method, which becomes even more clear as the dimensionality increases. Interestingly, the *worst*-performing function changes as dimensionality increases. In the 40-dimensional experiments, GP-UCB, which

generally performed well in other experiments, does quite poorly. Examining the behavior, it appears that by trying to minimize regret instead of maximizing improvement, GP-UCB favors regions of high variance. However, since a 40-dimensional space remains extremely sparsely populated even with hundreds of samples, the vast majority of the space still has high variance, and thus high acquisition value.

6.3.3 Control of a particle simulation

We also applied these methods to optimize the behavior of a simulated physical system in which the trajectories of falling particles are controlled via a set of repelling forces as introduced in Section 3.6.2. This is a difficult, nonlinear control task whose resulting objective function exhibits fairly isolated regions of high value surrounded by severe plateaus. We can briefly recap this problem as follows: the four-dimensional state-space in this problem consists of a particle's 2-dimensional position and velocity $(p, \dot{p})$ with two-dimensional actions consisting of forces which act on the particle. Particles are also affected by gravity and a frictional force resisting movement. The goal is to direct the path of the particle through regions of the state space with high reward $r(p)$ so as to maximize the *total reward* accumulated over many time-steps. In our experiments we use a finite, but large, time-horizon H. In order to control this system we employ a a set of "repellers", each of which is located at some position $c_i = (a_i, b_i)$ and has strength w_i. The force on a particle at position p is a weighted sum of the individual forces from all repellers, each of which is inversely proportional to the distance $p - c_i$.

This problem can be formulated in the setting of Bayesian optimization by defining the vector of repeller parameters $x = (w_1, a_1, b_1, \dots)$. In the experiments shown in Figure 6.5 we will utilize three repellers, resulting in a 9-dimensional optimization task. We can then define our objective as the total H-step expected reward $f(x) = \mathbb{E}\big[\sum_{t=1}^{H} r(p_n)|x\big]$. Finally, since the model defines a probability distribution $\Pr(p_{1:H}|x)$ over particle trajectories we can obtain a noisy estimate of this objective function by sampling a single trajectory and evaluating the sum over its immediate rewards.

Results for this optimization task are shown in Figure 6.5. As with the previous synthetic examples GP-Hedge outperforms each of its constituent methods. We further note the particularly poor performance of PI on this example, which in part we hypothesize is a result of plateaus in the resulting objective function. In particular PI has trouble exploring after it has "locked on" to a particular mode, a fact that seems exacerbated when there are large regions with very little change in objective. The figure also shows that gradient based methods, even when using smart tricks such as PEGASUS [Ng and Jordan, 2000], perform badly in comparison as the reward is severely multi-modal with large plateaus in between. Also note that we do not directly compare against the earlier reversible jump procedure of Chapter 3. Both the methods of this chapter and the methods of that chapter provide a method for performing gradient-free optimization, however we can see by comparing the time-scales that for time horizons of $H = 100$ that the time-scale necessary for convergence of these methods is orders of magnitude faster. This is a result of the MCMC methods of Chapter 3 requiring more samples and occasional proposal rejections in order to maintain detailed balance—something that may prove less important, and does so here, in the optimization setting.

6.4 Convergence behavior

Properly assessing the convergence behavior of hedging algorithms of this type is very problematic. The main difficulty lies with the fact that decisions made at iteration t affect the state of the problem and the resulting rewards at all future iterations. As a result we cannot relate the regret of our algorithm directly to the regret of the *best* underlying acquisition strategy: had we actually used the best underlying strategy we would have selected completely different points [Cesa-Bianchi and Lugosi, 2006, section 7.11].

Regret bounds for the underlying GP-UCB algorithm have been shown by [Srinivas et al., 2010]. Due to [Auer et al., 1998] we also have regret bounds for the hedging strategies used to select between acquisition functions (improved bounds can also be found in [Cesa-Bianchi and Lugosi,

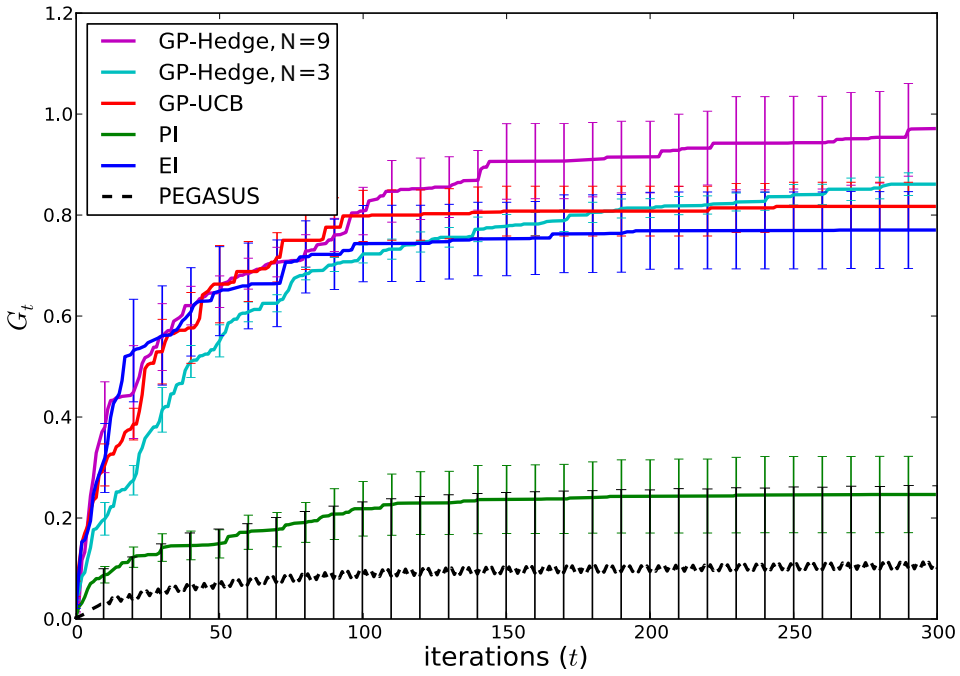

Figure 6.5: Results of experiments on the repeller control problem. The plot shows the progress of each of the described Bayesian optimization methods averaged over 25 runs. For comparison, it also shows the progress of a gradient method with PEGASUS.

2006]). However, because of the points stated in the previous paragraph, and expounded in more detail in the appendix, we cannot simply combine both regret bounds. See also the bounds for a related but distinct bandit problem of [Grunewalder et al., 2010].

With these caveats in mind we will consider a slightly different algorithmic framework. In particular we will consider rewards at iteration t given by the mean $r_n^{(i)} = \mu_{n-1}(x_n^{(i)})$, where we note that while this procedure still evaluates the GP mean, rewards are calculated before refitting. This assumption is made merely to simplify the following proof. We will also assume that GP-UCB is included as one of the possible acquisition functions due to its associated convergence results (see Section 6.1.2). In this scenario

we can obtain the following bound on our cumulative regret.

Theorem 1. *Assume GP-Hedge is used with a collection of acquisition strategies, one of which is GP-UCB with parameters β_n. If we also have a bound γ_n on the information gained at points selected by the algorithm after n iterations, then with probability at least $1-\delta$, for any finite horizon N the cumulative regret is bounded by*

$$L_N \leq \sqrt{NC_1\beta_N\gamma_N} + \Big[\sum_{n=1}^{N} \beta_n \sigma_{n-1}(x_n^{\mathrm{UCB}})\Big] + \mathcal{O}(\sqrt{N}),$$

where x_n^{UCB} is the nth point proposed by GP-UCB.

While we will give a full proof of this theorem in Section 6.4.1 we should note that this statement does not, on its own, guarantee the convergence of the algorithm, i.e. that $\lim_{N\to\infty} L_N/N = 0$. We can see, however, that our regret is bounded by two sub-linear terms and an additional term which depends on the information gained at points proposed, but not necessarily selected. In some sense this additional term depends on the proximity of points proposed by GP-UCB to points previously selected, the expected distance of which should decrease as the number of iterations increases.

We should point out, also, that the regret incurred by the hedging procedure is with respect to the *best underlying strategy*, which need not necessarily be GP-UCB. We then relate this *strategy regret* to the regret incurred by GP-UCB on the actual points proposed due to the known regret bounds for GP-UCB. An interesting extension to these ideas would be to incorporate bounds on the other underlying strategies, such as recent bounds for EI [Bull, 2011].

6.4.1 Proof of Theorem 1

We will consider a portfolio-based strategy using rewards $r_n^{(i)} = \mu_{n-1}(x_n^{(i)})$ and selecting between acquisition functions using the Hedge algorithm. In order to discuss this we will need to write the gain over N steps, in hindsight,

that would have been obtained had we instead used strategy i,

$$g_N^{(i)} = \sum_{n=1}^{N} r_n^{(i)} = \sum_{n=1}^{N} \mu_{n-1}(x_n^{(i)}).$$

We must emphasize however that this gain is conditioned on the actual decisions made by Hedge, namely that points $\{x_1, \ldots, x_{n-1}\}$ were selected by Hedge. If we define the maximum strategy $g_N^{\max} = \max_i g_N^{(i)}$ we can then bound the regret of Hedge with respect to this gain.

Lemma 1. *With probability at least* $1-\delta_1$ *and for a suitable choice of Hedge parameters,* $\eta = \sqrt{8 \ln M/N}$*, the regret is bounded by*

$$g_N^{\max} - g_N^{\text{Hedge}} \leq \mathcal{O}(\sqrt{T}).$$

This result is given without proof as it follows directly from [Cesa-Bianchi and Lugosi, 2006, Section 4.2] for rewards in the range $[0, 1]$. At the cost of slightly worsening the bound in terms of its multiplicative/additive constants, the following generalizations can also be noted:

- For rewards instead in the arbitrary range[2] $[a, b]$ the same bound can be shown by referring to [Cesa-Bianchi and Lugosi, 2006, Section 2.6].
- The choice of η in Lemma 1 requires knowledge of the time horizon N. By referring to [Cesa-Bianchi and Lugosi, 2006, Section 2.3] we can remove this restriction using a time-varying term $\eta_n = \sqrt{8 \ln M/n}$.
- By referring to [Cesa-Bianchi and Lugosi, 2006, Section 6.8] we can also extend this bound to the partial-information strategy Exp3.

Finally, we should also note that this regret bound trivially holds for any strategy i, since $g_N^{\max}$ is the maximum. It is also important to note that this lemma holds for any choice of $r_t^{(i)}$, *with rewards depending on the actual actions taken by Hedge.* The particular choice of rewards we use for this proof have been selected in order to achieve the following derivations.

For the next two lemmas we will refer the reader to [Srinivas et al., 2010,

[2] To obtain rewards bounded within some range $[a, b]$ we can assume that the additive noise ϵ_t is truncated above some large absolute value, which guarantees bounded means.

Lemma 5.1 and 5.3] for proof. The first provides a bound on deviations of the GP mean from the true function value, while the second allows us to rewrite the information gain for points selected in terms of the predictive variances. We point out, however, that these two lemmas only depend on the underlying Gaussian process and as a result can be used separately from the GP-UCB framework.

Lemma 2. *Assume $\delta_2 \in (0,1)$, a finite sample space $|\mathcal{A}| < \infty$, and $\beta_t = 2\log(|A|\pi_t/\delta_2)$ where $\sum_t \pi_t^{-1} = 1$ and $\pi_t > 0$. Then with probability at least $1-\delta_2$ the absolute deviation of the mean is bounded by*

$$|f(x) - \mu_{t-1}(x)| \leq \sqrt{\beta_t}\sigma_{t-1}(x) \quad \forall x \in A, \forall t \geq 1.$$

In order to simplify this discussion we have assumed that the sample space $\mathcal{A}$ is finite, however this can also be extended to compact spaces as in [Srinivas et al., 2010, Lemma 5.7].

Lemma 3. *The information gain for points selected by the algorithm can be written as*

$$I(y_{1:T}; f_{1:T}) = \frac{1}{2}\sum_{t=1}^{T} \log(1 + \sigma^{-2}\sigma_{t-1}^2(x_t)).$$

Next, the following lemma follows the proof of [Srinivas et al., 2010, Lemma 5.4], however again it can be applied outside the GP-UCB framework. Due to the slightly different conditions we recreate this proof here.

Lemma 4. *Given points x_n selected by the algorithm and for $C_1 = 2/\log(1+\sigma^{-2})$, the following bound holds for the sum of variances:*

$$\sum_{n=1}^{N} \beta_n \sigma_n^2(x_n) \leq C_1 \beta_N \gamma_N.$$

Proof. Because β_n is nondecreasing we can write the following inequality

$$\begin{aligned}\beta_n \sigma^2_{n-1}(x_n) &\leq \beta_N \sigma^2 (\sigma^{-2} \sigma^2_{n-1}(x_n)) \\ &\leq \beta_N \sigma^2 \frac{\sigma^{-2}}{\log(1+\sigma^{-2})} \log(1+\sigma^{-2}\sigma^2_{n-1}(x_n)).\end{aligned}$$

The second inequality holds because the posterior variance is bounded by the prior variance, $\sigma^2_{n-1}(x) \leq k(x,x) \leq 1$, which allows us to write

$$\sigma^{-2}\sigma^2_{n-1}(x_n) \leq \sigma^{-2} \frac{\log(1+\sigma^{-2}\sigma^2_{n-1}(x_n))}{\log(1+\sigma^{-2})}.$$

By summing over both sides of the original bound and applying the result of Lemma 3 we can see that

$$\begin{aligned}\sum_{n=1}^{N} \beta_n \sigma^2_{n-1}(x_n) &\leq \beta_N \frac{1}{2} C_1 \sum_{n=1}^{N} \log(1+\sigma^{-2}\sigma^2_{n-1}(x_n)) \\ &= \beta_N C_1 I(y_{1:N}; f_{1:N}).\end{aligned}$$

The result follows by bounding the information gain by $I(y_{1:N}; f_{1:N}) \leq \gamma_N$, which can be done for many common kernels, including the squared exponential [see Srinivas et al., 2010, Theorem 5]. □

Finally, the following lemma follows directly from [Srinivas et al., 2010, Lemma 5.2]. We will note that this lemma depends only on the definition of the GP-UCB acquisition function, and as a result does not require that points at any previous iteration were actually selected via GP-UCB.

Lemma 5. *If the bound from Lemma 2 holds, then for a point x_n^{UCB} proposed by GP-UCB with parameters β_n the following bound holds:*

$$f(x^*) - \mu_{n-1}(x_n^{\mathrm{UCB}}) \leq \sqrt{\beta_n}\sigma_{n-1}(x_n^{\mathrm{UCB}}).$$

We can now combine these results to construct the proof of Theorem 1.

Proof of Theorem 1. With probability at least $1-\delta_1$ the result of Lemma 1 holds. If we assume that GP-UCB is included as one of the acquisition

functions we can write

$$-g_N^{\text{Hedge}} \leq \mathcal{O}(\sqrt{N}) - g_N^{\text{UCB}}$$

and by adding $\sum_{n=1}^{N} f(x^*)$ to both sides this inequality can be rewritten as

$$\sum_{n=1}^{N} f(x^*) - \mu_{n-1}(x_n) \leq \mathcal{O}(\sqrt{N}) + \sum_{n=1}^{N} f(x^*) - \mu_{n-1}(x_n^{\text{UCB}}).$$

With probability at least $1 - \delta_2$ the bound from Lemma 2 can be applied to the left-hand-side and the result of Lemma 5 can be applied to the right. This allows us to rewrite this inequality as

$$\sum_{n=1}^{N} f(x^*) - f(x_n) - \sqrt{\beta_n}\sigma_{n-1}(x_n) \leq \mathcal{O}(\sqrt{N}) + \sum_{n=1}^{N} \sqrt{\beta_n}\sigma_{n-1}(x_n^{\text{UCB}})$$

which means that the regret is bounded by

$$\begin{aligned} L_N &= \sum_{n=1}^{N} f(x^*) - f(x_n) \\ &\leq \mathcal{O}(\sqrt{N}) + \sum_{n=1}^{N} \sqrt{\beta_n}\sigma_{n-1}(x_n^{\text{UCB}}) + \sum_{n=1}^{N} \sqrt{\beta_n}\sigma_{n-1}(x_n) \\ &\leq \mathcal{O}(\sqrt{N}) + \sum_{n=1}^{N} \sqrt{\beta_n}\sigma_{n-1}(x_n^{\text{UCB}}) + \sqrt{C_1 N \beta_N \gamma_N}. \end{aligned}$$

This final inequality follows directly from Lemma 4 by application of the Cauchy-Schwarz inequality. We should note that we cannot use Lemma 4 to further simplify the terms involving the sum over $\sigma_{n-1}(x_n^{\text{UCB}})$. This is because the lemma only holds for points that are sampled by the algorithm, which may not include those proposed by GP-UCB.

Finally, this result depends upon Lemmas 1 and 5 holding. By a simple union bound argument we can see that these both hold with probability at least $1 - \delta_1 - \delta_2$, and by setting $\delta_1 = \delta_2 = \delta/2$ we recover our result. □

6.5 Chapter summary and conclusions

Hedging strategies are a powerful tool in the design of acquisition functions for Bayesian optimization. In this paper we have shown that strategies that adaptively modify a portfolio of acquisition functions often perform substantially better—and almost never worse—than the best-performing individual acquisition function. This behavior was observed consistently across a broad set of experiments including high-dimensional GPs, standard test problems recommended in the bounded global optimization literature, and a hard continuous, 9D, nonlinear Markov decision process. These improvements will allow for advances in many practical domains of interest where we have already demonstrated the benefits of simple Bayesian optimization techniques [Brochu et al., 2010a,b, Martinez-Cantin et al., 2007], including robotics, online planning, hierarchical reinforcement learning, experimental design and interactive user interfaces.

Our experiments have also shown that full-information strategies are able to outperform partial-information strategies in many situations. However, partial-information strategies can be beneficial in instances of high N or in situations where the acquisition functions provide very conflicting advice. Evaluating these tradeoffs is an interesting area of future research.

In this work we give a regret bound for our hedging strategy by relating its performance to existing bounds for GP-UCB. Although our bound does not guarantee convergence it does provide some intuition as to the success of hedging methods in practice. Another interesting line of future research involves finding similar bounds for the gap measure.

Chapter 7

Conclusion

This work primarily focuses on probabilistic approaches to planning and reinforcement learning problems. In Chapter 2 we discussed an alternative probabilistic model whose maximum likelihood parameters correspond to the optimal parameters of an associated MDP. We then used this technique to propose a novel mixture-of-Gaussians MDP solution, as well as extended the probabilistic MDP formulation to semi-Markov decision processes. In Chapter 3 we extended this formalism to the Bayesian setting, allowing for the computation of a "posterior" over policy parameters. This also allowed for a novel gradient-free approach to the solution of direct policy search. In Chapter 4 we introduced a number of regularization schemes for LSTD, aimed at enforcing sparsity in the underlying value function approximation. Finally, in Chapters 5–6 we discussed Bayesian approaches to optimization, including bandit algorithms and Bayesian optimization. In this last chapter we continued to introduce a meta-optimization algorithm on top of various acquisition strategies which chooses between these lower level optimizers.

The results of these chapters significantly enhances the literature in the following ways:

- With the mixture-of-Gaussians model and its accompanying solution method we provided a proper, continuous extension to the planning-as-inference framework that is also analytically tractable. This extension also provides a solution method which generalized linear-quadratic

control to more general reward models.

- We were the first to extend this framework to the continuous-time, semi-Markov setting.
- We were the first to use MCMC (reversible-jump or otherwise) as a solution method for direct policy search.
- We provide a novel reformulation of the planning-as-inference framework in terms of the entire sum of rewards over a trajectory. This enhances the mixing properties of the Markov chain, but it also has applications to Monte Carlo EM; see also [Vlassis and Toussaint, 2009] where a similar approach was developed in parallel.
- We provide a novel decomposition of the state-space via auxiliary noise variables which greatly enhances the mixing properties of the Markov chain.
- We provide a novel regularization scheme for the problem of LSTD based on a mixture of ℓ_2 and ℓ_1 penalties, and show how it encourages sparse solutions, as well as allowing for solution methods that are much more in line with the approaches of standard supervised learning.
- We describe a novel meta-algorithm for performing Bayesian optimization by mixing over a set of acquisition functions; we also show that this method out-performs any single choice of acquisition function.
- Finally, we prove a bound on the performance of this strategy which relates its performance to the performance of the portfolio of underlying strategies.

This work also points towards a number of interesting avenues of future research. In Section 2.3 we presented an EM algorithm for solving a linear-Gaussian system with mixture-of-Gaussians rewards. This provides an analytic method for solving linear systems with rewards that are more general than standard quadratic cost functions. However, we can consider generalizing this further. For example, take the exponential family of dis-

tributions [see Wainwright and Jordan, 2008]

$$p(x|\eta) = h(x)\exp\{\eta^T T(x) - A(\eta)\}$$

for parameter vector η referred to as the *natural parameter* and functions h, T, and A. This family is quite general, encompassing Gaussian, binomial, multinomial, gamma, beta, and many other such distributions. As a result we can consider using transition models and rewards whose functional forms are given as conditional exponential family distributions. We can see that the Gaussian model of Section 2.3 is already a particular implementation of this more general idea of "exponential family MDPs". Further, if we consider sufficient statistics $T(x)$ which are linear, we should be able to employ efficient updates as alluded to earlier, and in [Furmston and Barber, 2011]. This would enable a unified approach with wide applicability to many different models.

Based on the earlier probabilistic interpretation of MDPs we further extended this methodology in Section 2.5 to semi-Markov decision problems. This allows us to tackle problems where time is continuous and variable for each state and action, but distributed according to some known distribution. This is an interesting problem in its own right, but it also allows one to consider hierarchical planning and reinforcement learning problems in the probabilistic paradigm. For example, we can consider a standard MDP with discrete-time actions, each of which takes a single unit of time to complete. But we can now consider *options* consisting of actions which correspond to *sub-policies* leading to a (possible random-length) sequence of individual actions; see [Ghavamzadeh and Mahadevan, 2007, Sutton et al., 1998]. If the sub-policies are known, this transforms the MDP into a SMDP as noted by Sutton et al.. The approach of Section 2.5 can be directly applied to this problem. However, we can also consider learning these sub-policies, which is not a problem we tackle in this work but is definitely of future interest.

In Chapter 3 we introduced an MCMC algorithm for sampling the parameters of an MDP proportional to the expected reward of trajectories sampled under those parameters. This is an interesting extension to the

purely maximum-likelihood approach of the previous chapter. This approach also provides a gradient-free method for optimization. However, in order to solve this problem using MCMC we must accept a certain number of rejection steps that are necessary to maintain detailed balance. Although this method gives us a full "posterior" distribution, this approach may not be necessary if one is only interested in optimization. However, consider the problem of model-based control wherein one estimates a model of the system from data, and then performs policy optimization given this model. This model itself is often subject to uncertainty due to the process by which it was learned. In this case we can envision a probabilistic model which takes into account both the uncertainty in the model and the "uncertainty" in the policy—i.e. the relative performance of the policy. We can then consider a mechanism which uses this policy "uncertainty" to direct further exploration of the model, ignoring areas which are unlikely to lead to significant increases in the expected reward.

Finally, in Chapter 5 we discussed a number of bandit methods for the tasks of both cumulative and simple regret optimization. In the first part of this chapter we discussed both Bayesian and frequentist approaches to the problem of minimizing frequentist regret, and we noted perhaps surprisingly that Bayesian approaches seemed to work quite well here, even though regret is a frequentist objective. This, of course, has been previously noted by [Kaufmann et al., 2012a]. We then introduced simple regret and a number of methods which minimize this objective, including a version of UCB, UCB-E, which is much more exploratory. However, we also saw that in many situations the cumulative strategies of the earlier sections also performed surprisingly well even under this different objective. Based on this observation, we hypothesize that a "Bayesian" version of UCB-E would perform quite well. This also has ramifications for the Bayesian optimization approaches of Chapter 6 which are inherently concerned with problems of minimizing simple regret.

Bibliography

A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1), 2008. → pages 90, 93, 94

H. Attias. Planning by probabilistic inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2003. → pages 20

J. Audibert and S. Bubeck. Best arm identification in multi-armed bandits. In *Proceedings of the Conference on Learning Theory*, 2010. → pages 131

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. Technical Report NC2-TR-1998-025, NeuroCOLT2 Technical Report Series, 1998. → pages 141, 152, 162

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2), 2002. → pages 122, 149

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the International Conference on Machine Learning*, 1995. → pages 13

D. Ballard and M. Hayhoe. Modelling the role of task in the control of gaze. *Visual Cognition*, 17(6-7), 2009. → pages 2

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15, 2001. → pages 15, 53

L. Bazzani, N. de Freitas, H. Larochelle, V. Murino, and J. Ting. Learning attentional policies for tracking and recognition in video with deep

networks. In *Proceedings of the International Conference on Machine Learning*, 2011. → pages 2

R. Bellman. A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics*, 16(3/4), 1956. → pages 115

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957. → pages 2, 8

D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995. → pages 11

C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006. → pages 50

S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996. → pages 4, 13, 89

E. Brochu. *Interactive Bayesian optimization: learning user preferences for graphics and animation*. PhD thesis, University of British Columbia, 2010. → pages 141, 155

E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems*, 2007. → pages 3, 141

E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2010a. → pages 3, 141, 144, 145, 169

E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv, 2010b. → pages 141, 143, 155, 169

S. Brooks, A. Gelman, G. Jones, and X. Meng, editors. *Handbook of Markov Chain Monte Carlo: Methods and Applications*. Chapman & Hall, 2010. → pages 177, 178

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Proceedings of the Conference on Algorithmic Learning Theory*, 2009. → pages 129, 130, 131

S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. §-armed bandits. *Journal of Machine Learning Research*, 2011. → pages 149

A. Bull. Convergence rates of efficient global optimization algorithms. eprint arXiv:1101.3501v2, arXiv, 2011. → pages 164

F. Bunea, A. Tsybakov, and M. Wegkamp. Sparsity oracle inequalities for the lasso. *Electronic Journal of Statistics*, 1, 2007. → pages 100

L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC Press, 2010. → pages 11

A. Cassandra, L. Kaelbling, and M. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the National Conference on Artificial Intelligence*, 1995. → pages 114

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, New York, 2006. → pages 162, 165

K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems*, 2009. → pages 152

K. Chaudhuri, Y. Freund, and D. Hsu. Tracking using an explanatory framework. Technical Report arXiv:0903.2862v2, arXiv, January 2010. → pages 141

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8), 1995. → pages 81

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6, 2005. → pages 143

D. Cox and S. John. SDO: A statistical method for global optimization. In M. Alexandrov and M. Hussaini, editors, *Multidisciplinary Design Optimization: State of the Art*, pages 315–329. SIAM, 1997. → pages 149

M. Crowley and D. Poole. Policy gradient planning for environmental decision making with existing simulators. In *AAAI11: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, Special Track on Computational Sustainability and AI*, 2011. → pages 2

P. Dayan and G. Hinton. Using EM for reinforcement learning. *Neural Computation*, 9, 1997. → pages 19

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1977. → pages 26

M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24(8), 2012. → pages 2

P. Diggle, A. Tawn, and A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society, Series C*, 47(3), 1998. → pages 143

L. Dixon and G. Szegö. The global optimization problem: an introduction. *Towards Global Optimization*, 2, 1978. → pages 155

A. Doucet, S. Godsill, and C. Robert. Marginal maximum a posteriori estimation using Markov chain Monte Carlo. *Statistics and Computing*, 12(1), 2002. → pages 79

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of statistics*, 32(2), 2004. → pages 103, 106

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the International Conference on Machine Learning*, 2003. → pages 13

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proceedings of the International Conference on Machine Learning*. ACM, 2005. → pages 13

A. Farahmand, M. Ghavamzadeh, C. Szepesvari, and S. Mannor. Regularized policy iteration. *Advances in Neural Information Processing Systems*, 21, 2009. → pages 90, 91, 96, 102

P. Fearnhead. State-space models. In Brooks et al. [2010]. → pages 68

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, 1995. → pages 152

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer, 2001. → pages 90, 98

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007. → pages 101, 104, 106

E. Frostig and G. Weiss. Four proofs of gittins multiarmed bandit theorem. *Applied Probability Trust*, 70, 1999. → pages 118

T. Furmston and D. Barber. Variational methods for reinforcement learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010. → pages 56

T. Furmston and D. Barber. Efficient inference in markov control problems. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2011. → pages 32, 56, 172

M. Geist and B. Scherrer. ℓ_1-penalized projected bellman residual. *European Workshop on Reinforcement Learning*, 2011. → pages 107

C. Geyer. Introduction to Markov Chain Monte Carlo. In Brooks et al. [2010]. → pages 60, 65

M. Ghavamzadeh and S. Mahadevan. Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research*, 8, 2007. → pages 55, 172

M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman. Finite-sample analysis of Lasso-TD. In *Proceedings of the International Conference on Machine Learning*, 2011. → pages iv, 17, 90, 101, 107

J. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, 41(2), 1979. → pages 16, 110, 116, 119, 138

J. Gittins and D. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3), 1979. → pages 115

J. Gittins, K. Glazebrook, and R. Weber. *Multi-armed bandit allocation indices*. Wiley, 2011. → pages 115, 116

P. Green. Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4), 1995. → pages 60, 61, 65, 66

P. Green. Trans-dimensional Markov Chain Monte Carlo. In P. Green, N. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*. Oxford University Press, 2003. → pages 59, 65

E. Greensmith, P. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2001. → pages 15

S. Grunewalder, J. Audibert, M. Opper, and J. Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010. → pages 163

L. Hannah and D. Dunson. Approximate dynamic programming for storage problems. In *Proceedings of the International Conference on Machine Learning*, 2011. → pages 2

N. Hay, S. Russell, D. Tolpin, and S. Shimony. Selecting computations: Theory and applications. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2012. → pages 133, 134

M. Hoffman and N. de Freitas. Inference strategies for solving semi-Markov decision processes. In L. Sucar, E. Morales, and H. Hoey, editors, *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*. IGI Global, 2011. → pages iv, 17

M. Hoffman, A. Doucet, N. de Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *Advances in Neural Information Processing Systems*, 2007a. → pages iv, 17

M. Hoffman, A. Doucet, N. de Freitas, and A. Jasra. On solving general state-space sequential decision problems using inference algorithms. Technical Report TR-2007-04, University of British Columbia, 2007b. → pages iv

M. Hoffman, P. Carbonetto, N. de Freitas, and A. Doucet. Inference strategies for solving semi-Markov decision processes. NIPS Workshop on Probabilistic Approaches for Robotics and Control, 2009a. → pages iv

M. Hoffman, N. de Freitas, A. Doucet, and J. Peters. An expectation maximization algorithm for continuous Markov decision processes with arbitrary reward. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009b. → pages iv, 17

M. Hoffman, H. Kück, N. de Freitas, and A. Doucet. New inference strategies for solving Markov decision processes using reversible jump MCMC. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2009c. → pages iv, 17

M. Hoffman, E. Brochu, and N. de Freitas. Portfolio strategies for Bayesian optimization. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2011a. → pages 18

M. Hoffman, A. Lazaric, M. Ghavamzadeh, and R. Munos. Regularized least squares temporal difference learning with nested ℓ_2 and ℓ_1 penalization. In *European Workshop on Reinforcement Learning*, 2011b. → pages iv, 17

D. Huang, T. Allen, W. Notz, and N. Zheng. Global optimization of stochastic black-box systems via sequential Kriging meta-models. *Journal of Global Optimization*, 3(34):441–466, March 2006. → pages 155

F. Hutter. *Automating the Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Vancouver, Canada, August 2009. → pages 141, 155

M. Jamshidian and R. Jennrich. Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, 88(421), 1993. → pages 30

J. Johns, C. Painter-Wakefield, and R. Parr. Linear complementarity for regularized policy evaluation and improvement. *Advances in Neural Information Processing Systems*, 23, 2010. → pages 90

D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79:157–181, 1993. → pages 146, 155

D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13: 455–492, 1998. → pages 140

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001. → pages 148

H. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *Cooperative Behavior in Neural Systems*, 2007. → pages 14, 20

E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2012a. → pages 124, 125, 127, 173

E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: an asymptotically optimal finite-time analysis. In *Proceedings of the Conference on Algorithmic Learning Theory*, 2012b. → pages 126

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems*, 2008. → pages 20

J. Kolter and A. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the International Conference on Machine Learning*, 2009. → pages 90, 91, 96, 101

H. Kück, M. Hoffman, A. Doucet, and N. de Freitas. Inference and learning for active sensing, experimental design, and control. In *Iberian Conference on Pattern Recognition and Image Analysis*, 2009. → pages 17

H. Kushner. A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86, 1964. → pages 146, 148

M. Lagoudakis and R. Parr. Model-free least-squares policy iteration. In *Advances in Neural Information Processing Systems*, 2002. → pages 13, 91, 102

T. Lai. Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, 15(3), 1987. → pages 122

T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6, 1985. → pages 121, 122, 124, 149

K. Lange. A quasi-Newton acceleration of the EM algorithm. *Statistica Sinica*, 5(1), 1995. → pages 30

F. Lewis, D. Dawson, and C. Abdallah. *Robot Manipulator Control: Theory and Practice*. CRC Press, 2004. → pages 41

D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2008. → pages 148, 155, 156, 158

D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007. → pages 141

H. Maei, C. Szepesvári, S. Bhatnagar, and R. Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the International Conference on Machine Learning*, 2010. → pages 13

O. Maron and A. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems*, 1994. → pages 132

R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. *Robotics: Science and Systems*, 2007. → pages 1, 141, 144, 169

R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2), 2009. → pages 1, 141

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1997. → pages 26, 30

T. Mitchell and J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404), 1988. → pages 59

V. Mnih, C. Szepesvári, and J. Audibert. Empirical bernstein stopping. In *Proceedings of the International Conference on Machine Learning*, 2008. → pages 133

J. Močkus, V. Tiesis, and A. Žilinskas. The application of bayesian methods for seeking the extremum. In L. Dixon and G. Szego, editors, *Toward Global Optimization*, volume 2. Elsevier, 1978. → pages 140, 148

P. Müller. Simulation based optimal design. In *Bayesian Statistics*, 1998. → pages 79

P. Müller, B. Sansó, and M. de Iorio. Optimal Bayesian design by inhomogeneous Markov chain simulation. *Journal of the American Statistical Association*, 99, 2004. → pages 79

A. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2000. → pages 38, 70, 78, 84, 162

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2000. → pages 2

A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics*, 2006. → pages 1

J. Niño-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing*, 23(2), 2011. → pages 119

J. Nocedal and S. Wright. *Numerical optimization*. Springer Verlag, 1999. ISBN 0387987932. → pages 146

M. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimization and Quadrature*. PhD thesis, University of Oxford, 2010. → pages 141, 145

O. Papaspiliopoulos, G. Roberts, and M. Sköld. Non-centered parameterisations for hierarchical models and data augmentation. *Bayesian Statistics*, 7, 2003. → pages 77

J. Peters and S. Schaal. Reinforcement learning for operational space control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007. → pages 1, 20

J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9), 2008. → pages 1, 15, 38

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2006. → pages 16

G. Poyiadjis, A. Doucet, and S. Singh. Maximum likelihood parameter estimation in general state-space models using particle methods. In *Proceedings of the American Statistical Association*, 2005. → pages 32

M. Puterman. *Markov Decision Processes*. Wiley-Interscience, 1994. → pages 2, 4, 8, 10, 11

M. Puterman and M. Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11), 1978. → pages 10

C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2004. → pages 13

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. → pages 144, 145, 156

C. E. Rasmussen. Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. *Bayesian Statistics*, 2003. → pages 141

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Robotics: Science and Systems*, 2012. → pages 20

S. Richardson and P. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B*, 59(4), 1997. → pages 60

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 55, 1952. → pages 2, 15, 109

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951. → pages 12, 14

C. Robert, G. Casella, and C. Robert. *Monte Carlo Statistical Methods*. Springer, 1999. → pages 60

C. Rothkopf and D. Ballard. Credit assignment in multiple goal embodied visuomotor behavior. *Frontiers in Psychology*, 1, 2010. → pages 2

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal Of The Royal Statistical Society Series B*, 71 (2), 2009. → pages 143

A. Sauré. *Approximate dynamic programming methods for advance patient scheduling*. PhD thesis, University of British Columbia, 2012. → pages 2

M. Schmidt. *Graphical Model Structure Learning with l1-Regularization*. PhD thesis, University of British Columbia, 2010. → pages 101, 104, 106

R. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 1988. → pages 19

E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2006. → pages 42

J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012. → pages 3

J. Spall. *Introduction to stochastic search and optimization: Estimation, simulation, and control*. Wiley-Interscience, 2005. → pages 78

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning*, 2010. → pages 125, 141, 149, 156, 159, 162, 165, 166, 167

R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1), 1988. → pages 12

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. → pages 4, 13

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1), 1998. → pages 172

R. Sutton, H. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the International Conference on Machine Learning*, 2009. → pages 13

C. Szepesvári. *Algorithms for reinforcement learning*. Morgan & Claypool Publishers, 2010. → pages 13

W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933. → pages 2, 125, 126

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1), 1996. → pages 100

E. Todorov. General duality between optimal control and estimation. In *IEEE Conference on Decision and Control*, 2008. → pages 20

M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the International Conference on Machine Learning*, 2006. → pages ii, 3, 19, 21, 32, 71

M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, University of Edinburgh, School of Informatics, 2006. → pages 20, 28, 31, 53

J. Tsitsiklis and B. Van Roy. Regression methods for pricing complex american-style options. *IEEE Transactions on Neural Networks*, 12(4), 2001. → pages 2

T. Ullman, C. Baker, O. Macindoe, O. Evans, N. Goodman, and J. Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Advances in Neural Information Processing Systems*, 2009. → pages 2

D. Verma and R. Rao. Planning and acting in uncertain environments using probabilistic inference. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2006. → pages 20

S. Vijayakumar, M. Toussaint, G. Petkos, and M. Howard. Planning and moving in dynamic environments: A statistical machine learning approach. In *Creating Brain Like Intelligence: From Principles to Complex Intelligent Systems*. Springer, 2009. → pages 20

N. Vlassis and M. Toussaint. Model-free reinforcement learning as mixture learning. In *Proceedings of the International Conference on Machine Learning*, 2009. → pages 29, 30, 75, 171

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008. → pages 56, 172

R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3), 1992. → pages 15

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2), 2005. → pages 97

Appendix A

Derivation of the Gaussian backward-message

We will first consider a reward and transition model parameterized as follows:

$$r(z) = \exp\Big[-\tfrac{1}{2}(c_k + z^T \Omega_k z - 2 z^T \mu_k)\Big],$$
$$f(z_n | z_{n-1}) = \mathcal{N}(z_n; F z_{n-1}, \Sigma).$$

In other words we are considering a single Gaussian reward. Extending the following derivations to mixture-of-Gaussian rewards is as simple as performing these updates for each component.

Next, we will assume that the backwards message at time n follows the same parameterization as the reward model, i.e. one given by (c_n, μ_n, Ω_n). Note that we will work directly with the standard time-indices for simplicity, but it is also possible to frame this in terms of a time-to-go τ. For this model we can recursively construct the β message at time $n-1$ as

$$\begin{aligned}
&\beta_{n-1}(z_{n-1}) \\
&\quad = \int \beta_n(z_n)\, f(z_n | z_{n-1})\, dz_n \\
&\quad = \int \exp\Big[-\tfrac{1}{2}(c_n + z_n^T \Omega_n z_n - 2 z_n^T \mu_n)\Big] \cdot |2\pi\Sigma|^{-1/2}
\end{aligned}$$

$$
\begin{aligned}
&\exp\Big[-\tfrac{1}{2}\big(z_n - Fz_{n-1}\big)^T\Sigma^{-1}\big(z_n - Fz_{n-1}\big)\Big]\,dz_n \\
&= \int \exp\Big[-\tfrac{1}{2}\Big\{c_n + z_n^T\Omega_n z_n - 2z_n^T\mu_n + \log|2\pi\Sigma| + \\
&\qquad\qquad z_n^T\Sigma^{-1}z_n + \big(Fz_{n-1}\big)^T\Sigma^{-1}\big(Fz_{n-1}\big) - 2z_n^T\Sigma^{-1}Fz_{n-1}\Big\}\Big]\,dz_n \\
&= \quad \exp\Big[-\tfrac{1}{2}\Big\{c_n + \log|2\pi\Sigma| + \big(Fz_{n-1}\big)^T\Sigma^{-1}\big(Fz_{n-1}\big)\Big\}\Big] \\
&\quad \underbrace{\int \exp\Big[-\tfrac{1}{2}\Big\{z_n^T\Omega_n z_n - 2z_n^T\mu_n + z_n^T\Sigma^{-1}z_n - 2z_n^T\Sigma^{-1}Fz_{n-1}\Big\}\Big]\,dz_n}_{(*)}.
\end{aligned}
$$

Here we have moved all the components not dependent on z_n outside the integral. The remaining terms inside the integral, marked with $(*)$, can then be computed as

$$
\begin{aligned}
&= \int \exp\Big[-\tfrac{1}{2}\Big\{z_n^T(\overbrace{\Omega_n + \Sigma^{-1}}^{\widetilde{\Sigma}^{-1}})x_n - 2z_n^T\widetilde{\Sigma}^{-1}\overbrace{\widetilde{\Sigma}(\mu_n + \Sigma^{-1}Fz_{n-1})}^{\widetilde{\mu}} \\
&\qquad\qquad + \widetilde{\mu}^T\widetilde{\Sigma}^{-1}\widetilde{\mu} - \widetilde{\mu}^T\widetilde{\Sigma}^{-1}\widetilde{\mu}\Big\}\Big] \\
&= \quad \exp\Big[-\tfrac{1}{2}\Big\{-\widetilde{\mu}^T\widetilde{\Sigma}^{-1}\mu\Big\}\Big]\int \exp\Big[-\tfrac{1}{2}\Big\{\big(z_n - \widetilde{\mu}\big)^T\widetilde{\Sigma}^{-1}\big(z_n - \widetilde{\mu}\big)\Big\}\Big]\,dz_n \\
&= \quad \exp\Big[-\tfrac{1}{2}\Big\{-\big(\mu_n + \Sigma^{-1}Fz_{n-1}\big)^T\widetilde{\Sigma}\widetilde{\Sigma}^{-1}\widetilde{\Sigma}\big(\mu_n + \Sigma^{-1}Fz_{n-1}\big)\Big\}\Big]\cdot|2\pi\widetilde{\Sigma}|^{1/2} \\
&= \quad \exp\Big[-\tfrac{1}{2}\Big\{\log|\tfrac{1}{2\pi}\widetilde{\Sigma}^{-1}| - \mu_n^T\widetilde{\Sigma}\mu_n - \big(Fz_{n-1}\big)^T\Sigma^{-1}\widetilde{\Sigma}\Sigma^{-1}\big(Fz_{n-1}\big) \\
&\qquad\qquad - 2(Fz_{n-1})^T\Sigma^{-1}\widetilde{\Sigma}\mu_n\Big\}\Big].
\end{aligned}
$$

This result can then be incorporated back into the original equation, and by expanding this term we arrive at

$$
\begin{aligned}
-2\log\beta_{n-1}(z_{n-1}) = c_n &+ \log|2\pi\Sigma| + z_{n-1}^T F^T\Sigma^{-1}Fz_{n-1} + \log|\tfrac{1}{2\pi}\widetilde{\Sigma}^{-1}| \\
&- \mu_n^T\widetilde{\Sigma}\mu_n - z_{n-1}^T F^T\Sigma^{-1}\widetilde{\Sigma}\Sigma^{-1}Fz_{n-1} \\
&- 2z_{n-1}^T F^T\Sigma^{-1}\widetilde{\Sigma}\mu_n.
\end{aligned}
$$

Grouping the terms that are constant, quadratic, and linear in z_{n-1} results

in

$$
\begin{aligned}
-2\log\beta_{n-1}(z_{n-1}) = &\left[c_n + \log|\Sigma\widetilde{\Sigma}^{-1}| - \big(\mu_n\big)^T\widetilde{\Sigma}\big(\mu_n\big)\right] \\
&+ \left[z_{n-1}^T F^T(\Sigma^{-1} - \Sigma^{-1}\widetilde{\Sigma}\Sigma^{-1})Fz_{n-1}\right] \\
&- \left[2z_{n-1}^T F^T\Sigma^{-1}(\widetilde{\Sigma}\mu_n)\right].
\end{aligned}
$$

Finally, by noting the earlier definition of

$$\widetilde{\Sigma}^{-1} = \Omega_n + \Sigma^{-1},$$

we can write the following updates for the backward message:

$$
\begin{aligned}
c_{n-1} &= c_n + \log|\Sigma\widetilde{\Sigma}^{-1}| - \mu_n^T\widetilde{\Sigma}\mu_n, \\
\Omega_{n-1} &= F^T(\Sigma^{-1} - \Sigma^{-1}\widetilde{\Sigma}\Sigma^{-1})F, \\
\mu_{n-1} &= F^T\Sigma^{-1}\widetilde{\Sigma}\mu_n.
\end{aligned}
$$