Bayesian Policy Gradient

Mohammad Ghavamzadeh MGH@CS.UALBERTA.CA
Yaakov Engel YAKI@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, CANADA

Abstract

Policy gradient methods are reinforcement learning algorithms that adapt a parameterized policy by following a performance gradient estimate. Many conventional policy gradient methods use Monte-Carlo techniques to estimate this gradient. The policy is improved by adjusting the parameters in the direction of the gradient estimate. Since Monte Carlo methods tend to have high variance, a large number of samples is required to attain accurate estimates, resulting in slow convergence. In this paper, we propose a Bayesian framework for policy gradient, by modeling the policy gradient as a Gaussian process. This reduces the number of samples needed to obtain accurate gradient estimates. Moreover, estimates of the natural gradient and the gradient covariance are provided at little extra cost. We perform experimental comparisons of the suggested algorithm with classic Monte-Carlo based algorithms on two simple problem domains.

1. Introduction

Policy Gradient (PG) methods[1] are Reinforcement Learning (RL) algorithms that maintain a parameterized action-selection policy and update the policy parameters by moving them in the direction of an estimate of the gradient of a performance measure. Early examples of PG algorithms are the class of REINFORCE algorithms of Williams (1992) which are suitable for solving problems in which the goal is to optimize the average reward. Subsequent work (e.g., Kimura et al., 1995; Marbach, 1998; Baxter & Bartlett, 2001) extended these algorithms to the cases of infinite-horizon MDPs and partially observable MDPs (POMDPs), and provided much needed theoretical analysis. However, both the theoretical analysis and empirical evaluations have highlighted a major shortcoming of these algorithms, namely, the high variance of the gradient estimates. This high variance may be traced to the fact that in most interesting cases, the time-average of the observed rewards is a high-variance (although unbiased) estimator of the true average reward, resulting in the extreme slowness and sample-inefficiency of these algorithms.

One solution proposed for this problem was to use a small (i.e., smaller than 1) "discount factor" in these algorithms (Marbach, 1998; Baxter & Bartlett, 2001), however, this creates another problem by introducing bias into the gradient estimates. Another solution, which does not involve biasing the gradient estimate, is to subtract a "reinforcement baseline" from the average reward estimate in the updates of PG algorithms (e.g., Williams, 1992; Marbach, 1998; Sutton et al., 2000). In Williams (1992) an average reward baseline was used, and in Sutton et al. (2000) it was conjectured that an approximate value function would be a good choice for a state-dependent baseline. However, in Weaver and Tao (2001) and Greensmith et al. (2004), it was shown, perhaps surprisingly, that the mean reward is in general *not* the optimal constant baseline, and that the true value function is generally *not* the optimal state-dependent baseline.

Another approach for speeding-up policy gradient algorithms was recently proposed in Kakade (2002) and refined and extended in Bagnell and Schneider (2003) and Peters et al. (2003). The idea is to replace the policy-gradient estimate with an estimate of the so-called *natural* policy-gradient. This is motivated by the requirement that the policy updates should be invariant to bijective transformations of the parametrization. Put more simply, a change in the

[1]The term has been coined in Sutton et al. (2000), but here we use it more liberally to refer to a whole class of reinforcement learning methods.

Appearing in *Kernel machines for reinforcement learning* workshop, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

way the policy is parametrized should not influence the result of the policy update. In terms of the policy update rule, the move to a natural-gradient rule amounts to linearly transforming the gradient using the inverse Fisher information matrix of the policy. In empirical evaluations, natural-PG has been shown to significantly outperform conventional PG (Kakade, 2002; Bagnell & Schneider, 2003).

However, both conventional and natural policy gradient methods rely on Monte-Carlo (MC) techniques to estimate the gradient of the performance measure. MC estimation is a purely frequentist procedure, and as such violates the *likelihood principle*[2] (Berger & Wolpert, 1984). Moreover, although MC estimates are unbiased, they make inefficient use of data, and therefore tend to produce estimates possessing high variance, or alternatively, require excessive sample sizes (see O'Hagan, 1987 for a discussion). In the case of policy-gradient estimation this is exacerbated by the fact that consistent policy improvement requires multiple gradient estimations.

In O'Hagan (1991) a Bayesian alternative to MC estimation was proposed.[3] The idea is to model integrals of the form $\int f(x)p(x)dx$ as Gaussian Processes (GPs). This is done by treating the first term f in the integrand as a *random function*, the randomness of which reflects our subjective uncertainty concerning its true identity. This allows us to incorporate our prior knowledge on f into its prior distribution. Observing (possibly noisy) samples of f at a set of points $(x_1, x_2, \dots, x_n)$ allows us to employ the Bayes' rule to compute a posterior distribution of f, conditioned on these samples. This, in turn, induces a posterior distribution over the values of the integral.

In this paper, we propose a Bayesian framework for policy gradient, by modeling the gradient as a Gaussian process. This reduces the number of samples needed to obtain accurate gradient estimates. Moreover, estimates of the natural gradient and the gradient covariance are provided at little extra cost. Additional gains may be attained by learning a transition model of the environment, allowing knowledge transfer between policies.

[2]The likelihood principle states that in a parametric statistical model, all the information about a data sample that is required for inferring the model parameters is contained in the likelihood function of that sample.

[3]O'Hagan (1991) mentions that this approach may be traced back even as far back as Poincaré (1896).

2. Reinforcement Learning and Policy Gradient Methods

Reinforcement Learning (RL) (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998) is a class of learning *problems* in which an agent (or controller) interacts with an unfamiliar, dynamic and stochastic environment (or plant), and whose goal is to maximize some measure of its long-term performance. This interaction is conventionally modeled as a Markov decision process (MDP), or, if the environmental state is not always completely observable, as a partially observable MDP (POMDP) (Puterman, 1994). In this work we restrict our attention to the discrete-time MDP setting.

Let $\mathcal{P}(\mathcal{X})$, $\mathcal{P}(\mathcal{A})$, and $\mathcal{P}(\mathbb{R})$ be the set of probability distributions on (Borel) subsets of $\mathcal{X}$, $\mathcal{A}$, and $\mathbb{R}$ respectively. A MDP is a tuple $(\mathcal{X}, \mathcal{A}, q, P, P_0)$ where $\mathcal{X}$ and $\mathcal{A}$ are the state and action spaces, respectively; $q(\cdot|x, a) \in \mathcal{P}(\mathbb{R})$ is the probability distribution over rewards received when action a is taken at state x; $P(\cdot|x, a) \in \mathcal{P}(\mathcal{X})$ is the probability distribution over next states, conditioned on action a being taken at state x (we assume that P and q are stationary); and $P_0 \in \mathcal{P}(\mathcal{X})$ is the probability distribution according to which the initial state is selected. We denote the random variable distributed according to $q(\cdot|x, a)$ as $r(x, a)$.

In addition, we need to specify the rule according to which the agent selects actions at each possible state. We assume that this rule does not depend explicitly on time. A *stationary policy* $\mu(\cdot|x) \in \mathcal{P}(\mathcal{A})$ is a probability distribution over actions, conditioned on the current state. Given a fixed policy μ, the sequence of state-action pairs (x_t, a_t) for $t = 0, 1, 2, \dots$ is generated by a Markov chain induced by that policy. The corresponding transition probability from (x_t, a_t) to (x_{t+1}, a_{t+1}) is $\mu(a_{t+1}|x_{t+1})P(x_{t+1}|x_t, a_t)$. We generically denote by $\xi = (x_0, a_0, x_1, a_1, \dots, x_{T-1}, a_{T-1}, x_T)$ a path generated by this Markov chain. The probability of such a path is given by

$$\Pr(\xi|\mu) = P_0(x_0) \prod_{t=0}^{T-1} \mu(a_t|x_t) P(x_{t+1}|x_t, a_t) \qquad (1)$$

We denote by $R(\xi)$ the (possibly discounted) *cumulative return* of the path ξ. $R(\xi) = \sum_{t=0}^{T-1} \gamma^t r(x_t, a_t)$ ($\gamma \in [0, 1]$ is a discount factor) is a random variable both because ξ is a random variable, and because, even for a given path ξ, each of the rewards collected in it may be stochastic. The expected value of $R(\xi)$ for a given path ξ is denoted by $\bar{R}(\xi)$. Finally, let us define the *expected return*

$$\eta(\mu) = \mathbf{E}(R(\xi)) = \int \bar{R}(\xi) \Pr(\xi|\mu) d\xi \qquad (2)$$

Gradient-based approaches to policy search in RL have recently received much attention as a means to sidetrack problems of partial observability and policy oscillations and even divergence encountered in value-function based methods (see Bertsekas & Tsitsiklis, 1996, Sections 6.4.2 and 6.5.3). In policy gradient (PG) methods, we define a class of smoothly parameterized stochastic policies $\{\mu(\cdot|x;\boldsymbol{\theta}), x \in \mathcal{X}, \boldsymbol{\theta} \in \Theta\}$, estimate the gradient of the expected return with respect to the policy parameters $\boldsymbol{\theta}$ from the observed system trajectories, and then improve the policy by adjusting the parameters in the direction of the gradient (Williams, 1992; Marbach, 1998; Baxter & Bartlett, 2001). The *score function* or *likelihood ratio* method has become the most prominent technique for gradient estimation via simulation. It has been first proposed in the sixties (Aleksandrov et al., 1968; Rubinstein, 1969) for computing performance gradients in i.i.d. (independently and identically distributed) processes, and was then extended to *regenerative* processes including MDPs by Glynn (1986; 1990), Reiman and Weiss (1986; 1989), Glynn and L'Ecuyer (1995), and to episodic MDPs by Williams (1992). This method estimates the gradient of the expected return

$$\eta(\boldsymbol{\theta}) = \eta(\mu(\cdot|\cdot;\boldsymbol{\theta})) = \int \bar{R}(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi, \tag{3}$$

with respect to the policy parameters $\boldsymbol{\theta}$. For that, we use the following equation:[4]

$$\nabla\eta(\boldsymbol{\theta}) = \int \bar{R}(\xi)\frac{\nabla\Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})}\Pr(\xi;\boldsymbol{\theta})d\xi, \tag{4}$$

where $\Pr(\xi;\boldsymbol{\theta}) = \Pr(\xi|\mu(\cdot|\cdot;\boldsymbol{\theta}))$, defined in Equation 1. In Equation 4, the quantity $\frac{\nabla\Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \nabla\log\Pr(\xi;\boldsymbol{\theta})$ is called the *score function* or *likelihood ratio*. Since the initial state distribution P_0 and the transition distribution P are independent of the policy parameters $\boldsymbol{\theta}$, we can write the likelihood ratio for a path ξ using Equation 1 as

$$\frac{\nabla\Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \sum_{t=0}^{T-1}\frac{\nabla\mu(a_t|x_t;\boldsymbol{\theta})}{\mu(a_t|x_t;\boldsymbol{\theta})} = \sum_{t=0}^{T-1}\nabla\log\mu(a_t|x_t;\boldsymbol{\theta})$$

Previous work on policy gradient used classical Monte-Carlo to estimate the gradient in Equation 4. These methods generate i.i.d. sample paths $\xi_1,\dots,\xi_M$ according to $\Pr(\xi;\boldsymbol{\theta})$, and estimate the gradient $\nabla\eta(\boldsymbol{\theta})$ using the following MC estimator:

$$\begin{aligned}\widehat{\nabla\eta}(\boldsymbol{\theta}) &= \frac{1}{M}\sum_{i=1}^{M}R(\xi_i)\nabla\log\Pr(\xi_i;\boldsymbol{\theta}) \\ &= \frac{1}{M}\sum_{i=1}^{M}R(\xi_i)\sum_{t=0}^{T_i-1}\nabla\log\mu(a_{t,i}|x_{t,i};\boldsymbol{\theta})\end{aligned} \tag{5}$$

This is an unbiased estimate and therefore, by the law of large numbers, $\widehat{\nabla\eta}(\boldsymbol{\theta}) \to \nabla\eta(\boldsymbol{\theta})$ with probability one.

3. Bayesian Quadrature

Bayesian quadrature (BQ) (O'Hagan, 1991) is, as its name suggests, a Bayesian method for evaluating an integral using samples of its integrand. We consider the problem of evaluating the integral

$$\rho = \int f(x)p(x)dx. \tag{6}$$

If $p(x)$ is a probability density function, this becomes the problem of evaluating the expected value of $f(x)$. A well known frequentist approach to evaluating such expectations is the Monte-Carlo method. For MC estimation of such expectations, it is typically required that samples $(x_1, x_2, \dots, x_M)$ are drawn from $p(x)$.[5] The integral in Equation 6 is then estimated as

$$\hat{\rho}_{MC} = \frac{1}{M}\sum_{i=1}^{M}f(x_i). \tag{7}$$

It is easy to show that $\hat{\rho}_{MC}$ is an unbiased estimate of ρ, with variance that diminishes to zero as $M \to \infty$. However, as O'Hagan (1987) points out, the MC estimation is fundamentally unsound, as it violates the likelihood principle, and moreover, does not make full use of the data at hand.

The alternative proposed in O'Hagan (1991) is based on the following reasoning: In the Bayesian approach, $f(\cdot)$ is random simply because it is numerically unknown. We are therefore uncertain about the value of $f(x)$ until we actually evaluate it. In fact, even then, our uncertainty is not always completely removed, since measured samples of $f(x)$ may be corrupted by noise. Modeling f as a GP means that our uncertainty is completely accounted for by specifying a Normal prior distribution over functions. This prior distribution is specified by its mean and covariance, and is denoted by $f(\cdot) \sim \mathcal{N}\{f_0(\cdot), k(\cdot,\cdot)\}$. This is

[4]Throughout the paper, we use the notation ∇ to denote $\nabla_{\boldsymbol{\theta}}$ – the gradient with respect to the policy parameters $\boldsymbol{\theta}$.

[5]If samples can only be drawn from some other distribution, importance sampling variants of MC can be used.

shorthand for the statement that f is a GP with prior mean and covariance

$$\mathbf{E}(f(x)) = f_0(x)\ ,\ \mathbf{Cov}(f(x), f(x')) = k(x, x'), \quad (8)$$

respectively. The choice of kernel function k allows us to incorporate prior knowledge on the smoothness properties of the integrand into the estimation procedure. When we are provided with a set of (possibly noisy) samples $\mathcal{D}_M = \{(x_i, y_i)\}_{i=1}^M$, where y_i is a sample of $f(x_i)$, we apply the Bayes' rule to condition the prior on these sampled values. The result is a Normal posterior distribution of $f|\mathcal{D}_M$. The expressions for the posterior mean and covariance are standard:

$$\begin{aligned} \mathbf{E}(f(x)|\mathcal{D}_M) &= f_0(x) + \boldsymbol{k}_M(x)^\top \boldsymbol{C}_M(\boldsymbol{y}_M - \boldsymbol{f}_0), \quad (9)\\ \mathbf{Cov}(f(x), f(x')|\mathcal{D}_M) &= k(x, x') - \boldsymbol{k}_M(x)^\top \boldsymbol{C}_M \boldsymbol{k}_M(x'). \end{aligned}$$

Here and in the sequel, we make use of the definitions:

$$\begin{aligned} \boldsymbol{f}_0 &= (f_0(x_1), \ldots, f_0(x_M))^\top, \\ \boldsymbol{y}_M &= (y_1, \ldots, y_M)^\top, \\ \boldsymbol{k}_M(x) &= (k(x_1, x), \ldots, k(x_M, x))^\top, \\ \boldsymbol{C}_M &= (\boldsymbol{K}_M + \boldsymbol{\Sigma}_M)^{-1}, \\ [\boldsymbol{K}_M]_{i,j} &= k(x_i, x_j), \end{aligned}$$

and $[\boldsymbol{\Sigma}_M]_{i,j}$ is the measurement noise covariance between the ith and jth samples. Typically, it is assumed that the measurement noise is i.i.d., in which case $\boldsymbol{\Sigma}_M = \sigma^2 \boldsymbol{I}$, where σ^2 is the noise variance and $\boldsymbol{I}$ is the (appropriately sized - here $M \times M$) identity matrix.

Since integration is a linear operation, the posterior distribution of the integral in Equation 6 is also Gaussian, and the posterior moments are given by (O'Hagan, 1991)

$$\begin{aligned} \mathbf{E}(\rho|\mathcal{D}_M) &= \int \mathbf{E}(f(x)|\mathcal{D}_M) p(x) dx, \quad (10)\\ \mathbf{Var}(\rho|\mathcal{D}_M) &= \iint \mathbf{Cov}(f(x), f(x')|\mathcal{D}_M) p(x) p(x') dx dx'. \end{aligned}$$

Substituting Equation 9 into Equation 10, we get

$$\begin{aligned} \mathbf{E}(\rho|\mathcal{D}_M) &= \rho_0 + \boldsymbol{z}_M^\top \boldsymbol{C}_M (\boldsymbol{y}_M - \boldsymbol{f}_0), \\ \mathbf{Var}(\rho|\mathcal{D}_M) &= z_0 - \boldsymbol{z}_M^\top \boldsymbol{C}_M \boldsymbol{z}_M, \quad (11) \end{aligned}$$

where we made use of the definitions:

$$\begin{aligned} \rho_0 &= \int f_0(x) p(x) dx \\ \boldsymbol{z}_M &= \int \boldsymbol{k}_M(x) p(x) dx \\ z_0 &= \iint k(x, x') p(x) p(x') dx dx'. \quad (12) \end{aligned}$$

Note that ρ_0 and z_0 are the prior mean and variance of ρ, respectively.

In Rasmussen and Ghahramani (2003) it has been experimentally demonstrated how this approach, when applied to the evaluation of an expectation, can outperform MC estimation by orders of magnitude, in terms of the mean-squared error.

In order to prevent the problem from "degenerating into infinite regress", as phrased by O'Hagan (1991), we should choose the functions p, k, and f_0 so as to allow us to solve the integrals in Equation 12 analytically. For instance, O'Hagan (1991) provides the analysis required for the case where the integrands in Equation 12 are products of multivariate Gaussians and polynomials, referred to as Bayes-Hermite quadrature. One of the contributions of the present paper is in providing analogous analysis for kernel functions that are based on the *Fisher kernel* (Jaakkola & Haussler, 1998; Shawe-Taylor & Cristianini, 2004).

It is important to note that in the MC estimation, samples must be drawn from the distribution $p(x)$, whereas in the Bayesian approach, samples may be drawn from arbitrary distributions. This affords us with flexibility in the choice of sample points, allowing us, for instance to actively design the samples $(x_1, x_2, \ldots, x_M)$ so as to maximize information gain.

4. Bayesian Policy Gradient

In this section, we use the Bayesian quadrature method to estimate the gradient of the expected return with respect to the policy parameters, and propose a new *Bayesian policy gradient* (BPG) algorithm.

In the frequentist approach to policy gradient, $\eta(\boldsymbol{\theta})$ from Equation 3 was our performance measure. In order to serve as a useful performance measure, it has to be a deterministic function of the policy parameters $\boldsymbol{\theta}$. This is achieved by averaging the cumulative return $R(\xi)$ over all possible paths ξ and all possible returns accumulated in each path. In the Bayesian approach we have an additional source of randomness, which is our subjective Bayesian uncertainty concerning the process generating the cumulative return. Let us denote

$$\eta_B(\boldsymbol{\theta}) = \int R(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi. \quad (13)$$

$\eta_B(\boldsymbol{\theta})$ is a random variable both because of the noise in $R(\xi)$ and the Bayesian uncertainty. Our deterministic Bayesian performance measure is therefore $\mathbf{E}(\eta_B(\boldsymbol{\theta})|\mathcal{D}_M)$, and a measure of the reliability of this measure is $\mathbf{Var}(\eta_B(\boldsymbol{\theta})|\mathcal{D}_M)$.

However, in this paper we are not interested in evaluating the posterior distribution of $\eta_B(\boldsymbol{\theta})$.[6] Instead, we are interested in optimizing performance, which is why we would rather evaluate the posterior distribution of the *gradient* of $\eta_B(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$:[7]

$$\nabla\eta_B(\boldsymbol{\theta}) = \int R(\xi)\frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})}\Pr(\xi;\boldsymbol{\theta})d\xi \tag{14}$$

In BPG, we cast the problem of estimating the gradient of the expected return (Equation 14) in the form of Equation 6. We therefore need to partition the integrand into two parts $f(\xi;\boldsymbol{\theta})$ and $p(\xi;\boldsymbol{\theta})$. We will place the GP prior over f and assume that p is known exactly. Because in general, $R(\xi)$ can not be known exactly, even for a given ξ (due to the stochasticity of the rewards), $R(\xi)$ should always belong to that part of the model upon which we place the GP prior ($f(\xi;\boldsymbol{\theta})$). Interestingly, in certain cases it is sufficient to know the Fisher information matrix corresponding to $\Pr(\xi;\boldsymbol{\theta})$, rather than having exact knowledge of $\Pr(\xi;\boldsymbol{\theta})$ itself. We make use of this fact in the sequel. We will then proceed by calculating the posterior moments of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ conditioned on the observed data. We investigate two different ways of partitioning the integrand in Equation 14 into a GP f and a function p below.

4.1. Model 1

In the first model, we define p and $\boldsymbol{f}$ as

$$\begin{aligned} p(\xi) &= \Pr(\xi;\boldsymbol{\theta}) \\ \boldsymbol{f}(\xi;\boldsymbol{\theta}) &= R(\xi)\frac{\nabla p(\xi;\boldsymbol{\theta})}{p(\xi;\boldsymbol{\theta})} = R(\xi)\nabla\log p(\xi;\boldsymbol{\theta}) \end{aligned}$$

We place a GP prior over $(\boldsymbol{f}(\xi_i;\boldsymbol{\theta}))_{i=1}^M$ which induces a GP prior over the corresponding noisy measurements $(\boldsymbol{y}_i)_{i=1}^M$

$$\begin{aligned} \boldsymbol{F}_M &= (\boldsymbol{f}(\xi_1;\boldsymbol{\theta}),\dots,\boldsymbol{f}(\xi_M;\boldsymbol{\theta})) \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}_M), \\ \boldsymbol{Y}_M &= (\boldsymbol{y}_1,\dots,\boldsymbol{y}_M) \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}_M+\sigma^2\boldsymbol{I}), \end{aligned}$$

where $\boldsymbol{K}_M$ is the kernel matrix. If n is the number of policy parameters, then, in this model, $\boldsymbol{f}(\xi;\boldsymbol{\theta})$ is an $n\times 1$ vector. Therefore, the i,jth element of the kernel matrix, $[\boldsymbol{K}_M]_{i,j}$ is in itself an $n\times n$ matrix which represents the covariance between the components of $\boldsymbol{f}(\xi_i;\boldsymbol{\theta})$ and $\boldsymbol{f}(\xi_j;\boldsymbol{\theta})$. For ease of exposition we assume $[\boldsymbol{K}_M]_{i,j} = k(\xi_i,\xi_j)\boldsymbol{I}$, which allows us to treat $[\boldsymbol{K}_M]_{i,j}$ as a scalar.

In this model, the posterior mean and covariance of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ are

$$\begin{aligned} \mathbf{E}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) &= \boldsymbol{Y}_M^\top\boldsymbol{C}_M\boldsymbol{z}_M, \\ \mathbf{Cov}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) &= (z_0 - \boldsymbol{z}_M^\top\boldsymbol{C}_M\boldsymbol{z}_M)\boldsymbol{I}, \end{aligned}$$

respectively, where

$$\begin{aligned} \boldsymbol{z}_M &= \int \boldsymbol{k}_M(\xi)\Pr(\xi;\boldsymbol{\theta})d\xi, \\ z_0 &= \iint k(\xi,\xi')\Pr(\xi;\boldsymbol{\theta})\Pr(\xi';\boldsymbol{\theta})d\xi d\xi'. \end{aligned}$$

A choice of kernel that allows us to derive closed form expressions for $\boldsymbol{z}_M$ and z_0 is the quadratic Fisher kernel (Jaakkola & Haussler, 1998):

$$k(\xi_i,\xi_j) = \left(1+\boldsymbol{u}(\xi_i)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_j)\right)^2, \tag{15}$$

where $\boldsymbol{u}(\xi) = \nabla\log\Pr(\xi;\boldsymbol{\theta})$ is the *score function* of the path ξ, and $\boldsymbol{G}$ is the Fisher information matrix defined as

$$\boldsymbol{G} = \mathbf{E}(\boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top) \tag{16}$$

Using this kernel, it is possible (albeit tedious) to show that $(\boldsymbol{z}_M)_i = 1+\boldsymbol{u}(\xi_i)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i)$ and $z_0 = 1+n$.

4.2. Model 2

In our second model, we define $\boldsymbol{p}(\xi) = \nabla\Pr(\xi;\boldsymbol{\theta})$ and $f(\xi) = \bar{R}(\xi)$. We place a GP prior over $(f(\xi_i;\boldsymbol{\theta}))_{i=1}^M$ which induces a GP prior over the corresponding noisy measurements $(y_i)_{i=1}^M$, where $y_i = R(\xi_i) = \bar{R}(\xi_i) + \left(R(\xi_i)-\bar{R}(\xi_i)\right)$. In this model, the posterior mean and variance of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ are

$$\begin{aligned} \mathbf{E}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) &= \boldsymbol{Z}_M\boldsymbol{C}_M\boldsymbol{y}_M, \\ \mathbf{Cov}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) &= \boldsymbol{Z}_0 - \boldsymbol{Z}_M\boldsymbol{C}_M\boldsymbol{Z}_M^\top, \end{aligned}$$

respectively, where

$$\begin{aligned} \boldsymbol{Z}_M &= \int \boldsymbol{k}_M(\xi)^\top\nabla\Pr(\xi;\boldsymbol{\theta})d\xi, \\ \boldsymbol{Z}_0 &= \iint k(\xi,\xi')\nabla\Pr(\xi;\boldsymbol{\theta})\nabla\Pr(\xi';\boldsymbol{\theta})^\top d\xi d\xi'. \end{aligned}$$

A choice of kernel that allows us to derive closed form expressions for $\boldsymbol{Z}_M$ and $\boldsymbol{Z}_0$ is the Fisher kernel (Jaakkola & Haussler, 1998):

$$k(\xi_i,\xi_j) = \boldsymbol{u}(\xi_i)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_j) \tag{17}$$

Using this kernel, $\boldsymbol{Z}_M = \boldsymbol{U}_M$ and $\boldsymbol{Z}_0 = \boldsymbol{G} - \boldsymbol{U}_M\boldsymbol{C}_M\boldsymbol{U}_M^\top$, where $\boldsymbol{U}_M = \left[\boldsymbol{u}(\xi_1)\ ,\ \boldsymbol{u}(\xi_2)\ ,\ \dots\ ,\ \boldsymbol{u}(\xi_M)\right]$.

[6] Although this is an interesting question in its own right.

[7] Note that, whereas $\nabla\mathbf{E}(\eta_B(\boldsymbol{\theta})) = \mathbf{E}(\nabla\eta_B(\boldsymbol{\theta}))$, the same is not true for the variance. That is, $\nabla\mathbf{Var}(\eta_B(\boldsymbol{\theta})) \neq \mathbf{Cov}(\nabla\eta_B(\boldsymbol{\theta}))$.

4.3. A Bayesian Policy Gradient Evaluation Algorithm

We can now use Model 1 and Model 2 to define algorithms for evaluating the gradient of the expected return with respect to the policy parameters. Pseudocode for these algorithms is shown in Algorithm 1. The generic algorithm (for either model) takes a set of policy parameters $\boldsymbol{\theta}$ and a sample size M as input, and returns an estimate of the gradient of the expected return with respect to the policy parameters $\nabla \eta_B(\boldsymbol{\theta})$. This algorithm generates M sample paths to evaluate the gradient. For each path ξ_i, the algorithm first computes its score function $\boldsymbol{u}(\xi_i)$ (Line 6). The score function is needed for computing the kernel function k, the measurement $\boldsymbol{y}$ in Model 1, and $\boldsymbol{z}$ or $\boldsymbol{Z}$. The algorithm then computes the return R and the measured y for the observed path ξ (Lines 7 and 8), and expands the kernel matrix $\boldsymbol{K}$ using

$$\boldsymbol{K}_i = \begin{bmatrix} \boldsymbol{K}_{i-1} & \boldsymbol{k}_{i-1}(\xi_i) \\ \boldsymbol{k}_{i-1}(\xi_i)^\top & k(\xi_i, \xi_i) \end{bmatrix} \tag{18}$$

Finally, the algorithm adds the measurement error $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$ to the covariance matrix $\boldsymbol{K}$ (Line 12), and computes the posterior mean for the gradient (Line 13). $\boldsymbol{Z}(:, i)$ on Line 10 represents the ith column of matrix $\boldsymbol{Z}$.

Algorithm 1 A Bayesian Policy Gradient Evaluation Algorithm

```
1: BPG_Eval($\boldsymbol{\theta}, M$)
       • sample size $M > 0$
       • a set of policy parameters $\boldsymbol{\theta} \in \mathbb{R}^n$
2: Set $\boldsymbol{G} = \boldsymbol{G}(\boldsymbol{\theta})$ ,  $\mathcal{D}_0 = \emptyset$
3: for $i = 1$ to $M$ do
4:     Sample a path $\xi_i$ using the policy $\mu(\boldsymbol{\theta})$
5:     $\mathcal{D}_i = \mathcal{D}_{i-1} \bigcup$     /*$\xi_i$*/
6:     Compute $\boldsymbol{u}(\xi_i) = \sum_{t=0}^{T_i-1} \nabla \log \mu(a_{t,i}|s_{t,i}; \boldsymbol{\theta})$
7:     Compute $R(\xi_i) = \sum_{t=0}^{T_i-1} r(s_{t,i}, a_{t,i})$
8:     Compute
           $\boldsymbol{y}(\xi_i) = R(\xi_i)\boldsymbol{u}(\xi_i)$          (Model 1)
         or
           $y(\xi_i) = R(\xi_i)$                    (Model 2)
9:     Compute $\boldsymbol{K}_i$ using $\boldsymbol{K}_{i-1}$ and $\xi_i$
10:    Compute
           $\boldsymbol{z}_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)$      (Model 1)
         or
           $\boldsymbol{Z}(:, i) = \boldsymbol{u}(\xi_i)$              (Model 2)
11: end for
12: Compute $\boldsymbol{C} = (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}$
13: Compute the posterior mean
         $\mathbf{E}(\nabla \eta_B(\boldsymbol{\theta})|\mathcal{D}_M) = \boldsymbol{Y}^\top \boldsymbol{C} \boldsymbol{z}$      (Model 1)
       or
         $\mathbf{E}(\nabla \eta_B(\boldsymbol{\theta})|\mathcal{D}_M) = \boldsymbol{Z} \boldsymbol{C} \boldsymbol{y}$      (Model 2)
14: return $\mathbf{E}(\nabla \eta_B(\boldsymbol{\theta})|\mathcal{D}_M)$
```

The kernel functions used in Model 1 and Model 2 (Equations 15 and 17) are both based on the Fisher kernel. Computing the Fisher kernel requires calculating the Fisher information matrix $\boldsymbol{G}$ (Equation 16). Therefore, we need to calculate the Fisher information matrix every time we update the policy parameters. In Algorithm 1 we assume that the Fisher information matrix is available, and leave its estimation for future work.

Nonetheless, let us briefly outline three possible methods for estimating the Fisher information matrix in an online manner:

1) MC Estimation: At each step j, our BPG algorithm generates M sample paths using the current policy parameters $\boldsymbol{\theta}_j$ in order to estimate the gradient $\nabla \eta_B(\boldsymbol{\theta}_j)$. We can use these generated sample paths to estimate the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta}_j)$ in a MC fashion as

$$\hat{\boldsymbol{G}}_{MC}(\boldsymbol{\theta}_j) = \frac{1}{\sum_{i=1}^M T_i} \sum_{i=1}^M \sum_{t=0}^{T_i-1} \nabla \log \mu(a_{t,i}|x_{t,i}; \boldsymbol{\theta}_j) \nabla \log \mu(a_{t,i}|x_{t,i}; \boldsymbol{\theta}_j)^\top.$$

$\hat{\boldsymbol{G}}_{MC}$ is an unbiased frequentist estimator of $\boldsymbol{G}$.

2) Bayesian Estimation: The Fisher information matrix is the result of an expectation integral (Equation 16). Therefore, it can be estimated using the Bayesian quadrature method in the same way that we estimate the gradient in this paper. We place a prior on the integrand in Equation 16, observe sample paths, and compute the posterior of the integrand which in turn implies a distribution over Fisher information matrices.

3) Model-Based Policy Gradient: The Fisher information matrix depends on the probability distribution over paths. This distribution is a product of two factors, one corresponding to the current policy, and the other corresponding to the MDP dynamics P(see Equation 1). Thus, if the MDP dynamics are known, the Fisher information matrix can be evaluated offline. We can model the MDP dynamics using some parameterized model, and estimate the model parameters using maximum likelihood or Bayesian methods. This would be a model-based approach to policy gradient, which might be an interesting research direction, since it would allow us to transfer information between different policies. Current policy gradient algorithms, including the algorithms described in this paper, are extremely wasteful of training data, since they do not have any *disciplined* way to use data collected for previous policy updates in computing the update of the current policy.

4.4. A Bayesian Policy Gradient Algorithm

Pseudo-code for the Bayesian policy gradient (BPG) algorithm is shown in Algorithm 2. This algorithm starts with an initial vector of policy parameters $\boldsymbol{\theta}_0$ and updates the parameters in the direction of the estimated gradient of the expected return. This is repeated N times. A more reasonable way to terminate the algorithm is to stop when the estimated gradient is nearly zero. For each parameter update, this algorithm calls the Bayesian policy gradient evaluation algorithm **BPG_Eval** described in Algorithm 1.

Algorithm 2 A Bayesian Policy Gradient Algorithm

```
1: BPG($\alpha, N, M$)
       • learning rate $\alpha_j$ , $j = 0, \ldots, N-1$
       • number of policy updates $N > 0$
       • sample size for gradient evaluation $M > 0$
2: Set $\boldsymbol{\theta}_0$
3: for $j = 0$ to $N-1$ do
4:     $\Delta\boldsymbol{\theta}_j =$ BPG_Eval($\boldsymbol{\theta}_j, M$)
5:     $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j + \alpha_j \Delta\boldsymbol{\theta}_j$          (Regular Gradient)
         or
       $\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j + \alpha_j \boldsymbol{G}^{-1}\Delta\boldsymbol{\theta}_j$      (Natural Gradient)
6: end for
7: return $\boldsymbol{\theta}_N$
```

4.5. Online Sparsification

In order to make Algorithm 1 more practical, we need to reduce the computational burden associated with the computation of the gradient. To do so, we incorporate online sparsification into the algorithm. This allows us to significantly reduce the time and space complexity of the algorithm, and ensures that matrix inversions are numerically stable.

We use the online sparsification method from Engel et al. (2002) (see also Csató & Opper, 2002) to selectively add a new observed path to the set of *dictionary* paths $\tilde{\mathcal{D}}_M$. We only add the new path ξ_i to $\tilde{\mathcal{D}}$, if $\delta_i = k(\xi_i, \xi_i) - \tilde{\boldsymbol{k}}_{i-1}(\xi_i)^\top \tilde{\boldsymbol{K}}_{i-1}^{-1} \tilde{\boldsymbol{k}}_{i-1}(\xi_i) > \lambda$. If the new path is added to $\tilde{\mathcal{D}}$ the dictionary kernel matrix $\tilde{\boldsymbol{K}}$ is expanded as shown in Equation 18. λ is a positive threshold parameter that determines the level of accuracy of the approximation as well as the level of sparsity attained.

Using this sparsification method, the posterior mean of the gradient is computed for Model 1 and Model 2 as

$$\mathbf{E}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) = \boldsymbol{Y}_M^\top \boldsymbol{A}_M (\boldsymbol{A}_M^\top \boldsymbol{A}_M)^{-1} \tilde{\boldsymbol{C}}_M \tilde{\boldsymbol{z}}_M,$$
$$\mathbf{E}(\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M) = \tilde{\boldsymbol{Z}}_M \tilde{\boldsymbol{C}}_M (\boldsymbol{A}_M^\top \boldsymbol{A}_M)^{-1} \boldsymbol{A}_M^\top \boldsymbol{y}_M,$$

respectively. $\tilde{\boldsymbol{C}}_M = \left(\tilde{\boldsymbol{K}}_M + \sigma^2 \boldsymbol{I}\right)^{-1}$ is an $m \times m$ matrix, where $m \leq M$ is the cardinality of $\tilde{\mathcal{D}}_M$. The matrix $\boldsymbol{A}_M$ is an $M \times m$ matrix whose ith row is $[\boldsymbol{A}_M]_{i,|\tilde{\mathcal{D}}_i|} = 1$ and $[\boldsymbol{A}_M]_{i,j} = 0$; $\forall j \neq |\tilde{\mathcal{D}}_i|$, if we add the sample path ξ_i to the set of sample paths, and is $\tilde{\boldsymbol{k}}_{i-1}(\xi_i)^\top \tilde{\boldsymbol{K}}_{i-1}^{-1}$ followed by zeros otherwise. Finally, $(\tilde{\boldsymbol{z}}_M)_i = 1 + \boldsymbol{u}(\xi_i)^\top \boldsymbol{G}^{-1} \boldsymbol{u}(\xi_i)$ and $\tilde{\boldsymbol{Z}}_M = [\boldsymbol{u}(\xi_1)\ ,\ \boldsymbol{u}(\xi_2)\ ,\ \ldots\ ,\ \boldsymbol{u}(\xi_M)]$ with $\xi_i \in \tilde{\mathcal{D}}_M$.

5. Experimental Results

In this section, we compare the Bayesian quadrature (BQ) and the simple MC gradient estimations using a simple bandit problem as well as a continuous state and action linear quadratic regulator (LQR). We also evaluate the performance of the Bayesian policy gradient (BPG) algorithm described in Algorithm 2 on the LQR, and compare it with a MC-based policy gradient (MCPG) algorithm.

5.1. A Simple Bandit Problem

The goal of this simple example is to compare the BQ and MC estimates of the gradient (for a fixed set of policy parameters) using the same samples. Our simple bandit problem has one state and $a \in \mathcal{A} = \mathbb{R}$. Thus, each path consists of one action. The policy is Gaussian with mean zero and variance one. As a result the probability of a path is also Gaussian with the same mean and variance: $\Pr(\xi) = \mu(a|x) \sim \mathcal{N}(0,1)$. The score function of the path $\xi = a$ and the Fisher information matrix $\boldsymbol{G}$ are computed as follows:

$$\nabla \log \Pr(\xi) = \begin{pmatrix} a \\ a^2 - 1 \end{pmatrix} \quad ; \quad \boldsymbol{G} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Table 1 shows the exact gradient of the expected return and its MC and BQ estimates (using 10 and 100 samples) for two versions of the simple bandit problem corresponding to two different reward functions $r(a) = a$ and $r(a) = a^2$. The average over 10^4 runs of the MC and BQ estimates and their standard deviations are reported in Table 1. The gradient is analytically computable in this problem and is reported as "Exact" in Table 1 for comparison purposes.

As shown in Table 1, the BQ estimate has much lower standard deviation than the MC estimate for both small and large sample sizes. The BQ estimate has

	$r(a) = a$	$r(a) = a^2$
Exact	$\begin{pmatrix}1\\0\end{pmatrix}$	$\begin{pmatrix}0\\2\end{pmatrix}$
MC (10)	$\begin{pmatrix}0.9950 \pm 0.438\\-0.0011 \pm 0.977\end{pmatrix}$	$\begin{pmatrix}0.0136 \pm 1.246\\2.0336 \pm 2.831\end{pmatrix}$
BQ (10)	$\begin{pmatrix}0.9856 \pm 0.050\\0.0006 \pm 0.060\end{pmatrix}$	$\begin{pmatrix}0.0010 \pm 0.082\\1.9250 \pm 0.226\end{pmatrix}$
MC (100)	$\begin{pmatrix}1.0004 \pm 0.140\\0.0040 \pm 0.317\end{pmatrix}$	$\begin{pmatrix}0.0051 \pm 0.390\\1.9869 \pm 0.857\end{pmatrix}$
BQ (100)	$\begin{pmatrix}1.0000 \pm 0.000001\\0.0000 \pm 0.000004\end{pmatrix}$	$\begin{pmatrix}0.0000 \pm 0.000003\\2.0000 \pm 0.000011\end{pmatrix}$

Table 1. The true gradient of the expected return and its MC and BQ estimates for two versions of the simple bandit problem corresponding to two different reward functions.

also better mean than the MC estimate for the large sample size ($M = 100$), and almost the same mean for the small sample size ($M = 10$).

5.2. Linear Quadratic Regulator

In this section, we consider the following linear system:

System:
Initial State: $x_0 \sim \mathcal{N}(0.3, 0.001)$
State Transition: $x_{t+1} = x_t + a_t + n_x$; $n_x \sim \mathcal{N}(0, 0.01)$
Reward: $r_t = x_t^2 + 0.1a_t^2$

Policy:
Actions: $a_t \sim \mu(a_t|x_t; \boldsymbol{\theta}) = \mathcal{N}(cx_t, \sigma^2)$
Parameters: $\boldsymbol{\theta} = (c\ ,\ \sigma)^\top$

The goal here is to maximize the expected return over 20 steps. Thus, it is an episodic problem with paths of length 20.[8] We run two sets of experiments on this system. We first fix the set of policy parameters and compare the BQ and MC estimates of the gradient of the expected return using the same samples. We then proceed to solving the complete policy gradient problem, and compare the performance of the BPG algorithm (with both regular and natural gradients) with a MC-based policy gradient (MCPG) algorithm.

5.2.1. Gradient Estimation

In this section, we compare the BQ and MC estimates of the gradient of the expected return for the policy represented by parameters $c = -0.2$ and $\sigma = 1$ using the same samples. We use both Model 1 (with and without sparsification) and Model 2 to compute the BQ estimates in this problem. We use several different sample sizes (number of paths used for gradient estimation) $M = 5j$, $j = 1, \ldots, 20$ for the BQ and MC estimates. For each sample size, we run the MC and BQ estimators 10^4 times, and then take the average. The true gradient is estimated using MC with 10^7 sample paths, for comparison purposes.

[8]Here, each path consists of 41 elements, 21 states $x_0, \ldots, x_{20}$ and 20 actions $a_0, \ldots, a_{19}$.

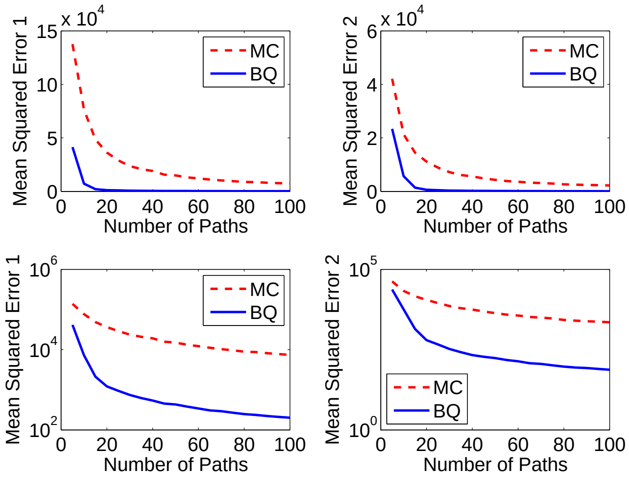

Figure 1. Results for the LQR using Model 1 without sparsification. Shown are the MSE of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M in linear (top row) and logarithmic (bottom row) scales.

Figure 1 shows the mean squared error (MSE) of the MC and BQ gradient estimates for different sample sizes in both linear (top row) and logarithmic (bottom row) scales. Figure 2 shows the mean and variance (over 10^4 runs) of the MC and BQ gradient estimations for different sample sizes. Note that although in the first row of Figure 2 the mean of the MC estimate is better than the mean of the BQ estimate for the small sample sizes (5 and 10), the MSE of the MC estimate is still worse than the MSE of the BQ estimate (Figure 1). This is due to the higher variance of the MC estimates shown in the bottom row of Figure 2. Figure 3 shows the mean absolute angular error of the MC and BQ estimates of the gradient for several different sample sizes. The absolute angular error is the absolute value of the angle between the true gradient and the estimated gradient. For these figures, the BQ gradient estimate was calculated using Model 1 without sparsification.

Figures 4, 5, and 6 show the mean squared error, the mean and the variance of the gradient estimates, and the mean absolute angular error for the BQ and MC estimation methods, when the BQ gradient estimation is computed using Model 1 with sparsification. As seen in these figures, sparsification slightly reduces the performance of the BQ gradient estimation. However, it makes the BQ method much faster and more

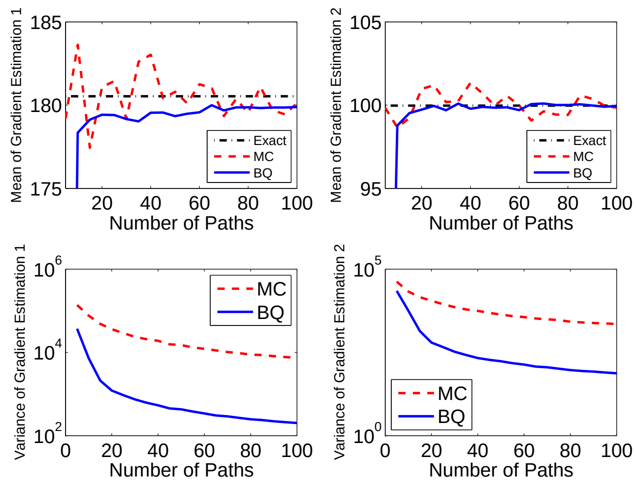

Figure 2. Results for the LQR using Model 1 without sparsification. Shown are the mean and variance of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M.

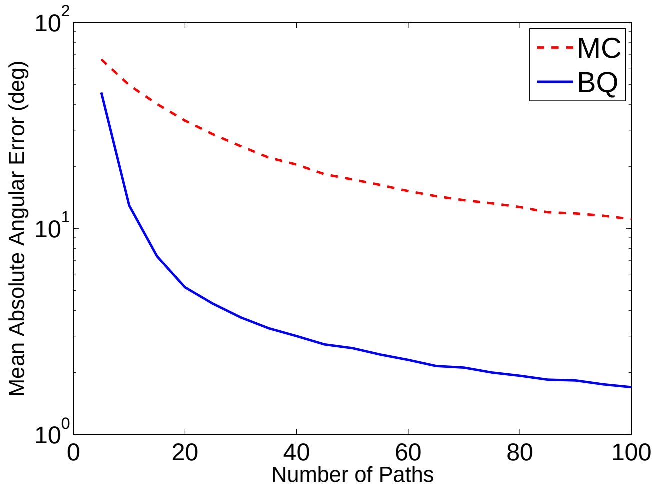

Figure 3. Results for the LQR using Model 1 without sparsification. Shown is the mean absolute angular error of the MC and BQ gradient estimations as a function of the number of sample paths M.

efficient, especially for large sample sizes. To give an intuition about the speed and the efficiency attained by sparsification, we should mention that the dimension of the feature space for the kernel used in Model 1 is 6 (Proposition 9.2 in Shawe-Taylor & Cristianini, 2004). Therefore, we deal with a kernel matrix of size 6 with sparsification versus a kernel matrix of size $M = 5j$, $j = 1, \ldots, 20$ without sparsification.

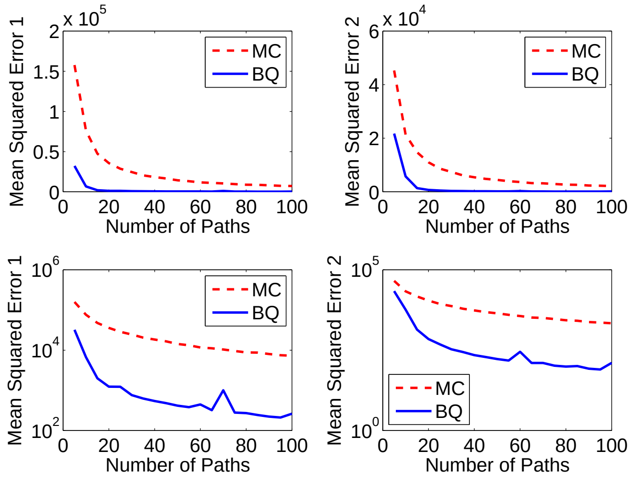

Figure 4. Results for the LQR using Model 1 with sparsification. Shown are the MSE of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M in linear (top row) and logarithmic (bottom row) scales.

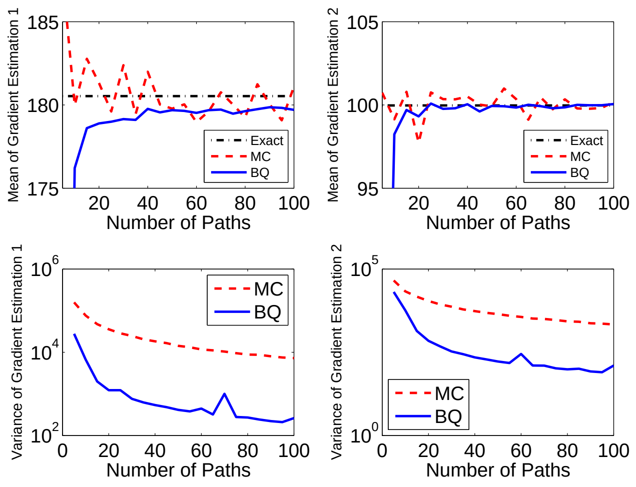

Figure 5. Results for the LQR using Model 1 with sparsification. Shown are the mean and variance of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M.

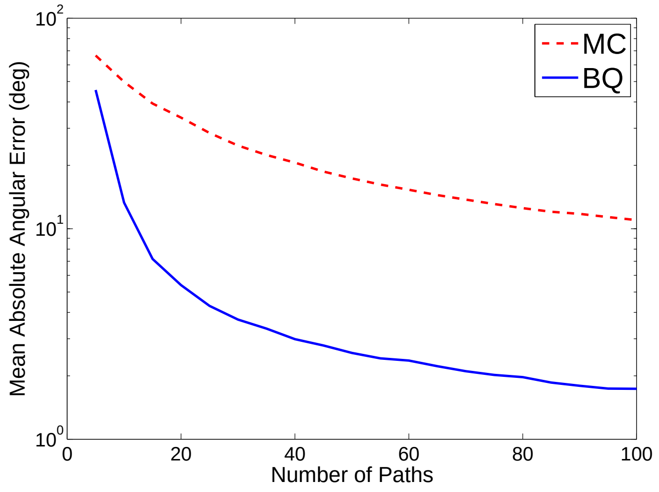

Figure 6. Results for the LQR using Model 1 with sparsification. Shown is the mean absolute angular error of the MC and BQ gradient estimations as a function of the number of sample paths M.

Figures 7, 8, and 9 show analogous results for Model 2 without sparsification. As usual the mean squared error, the mean and the variance of the gradient estimates, and the mean absolute angular error for the BQ and MC estimation methods are shown in these figures.

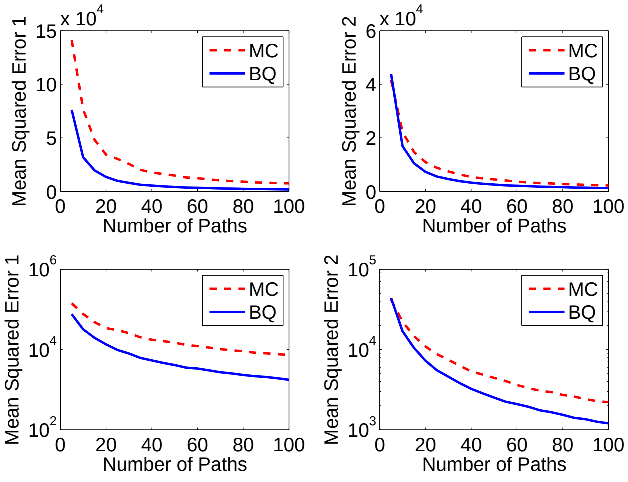

Figure 7. Results for the LQR using Model 2 without sparsification. Shown are the MSE of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M in linear (top row) and logarithmic (bottom row) scales.

We now add i.i.d. Gaussian noise to the reward func-

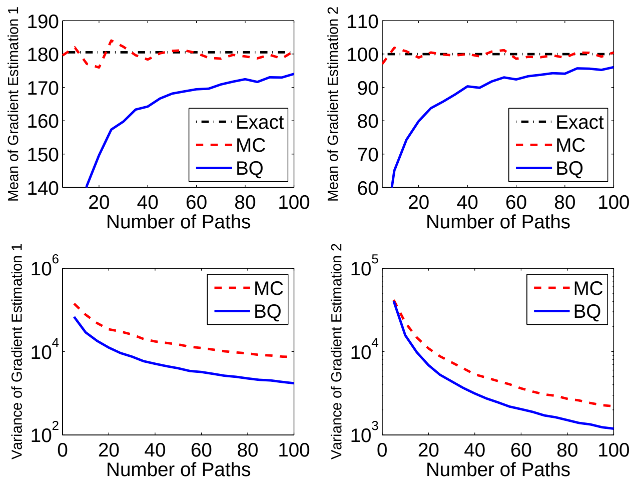

Figure 8. Results for the LQR using Model 2 without sparsification. Shown are the mean and variance of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M.

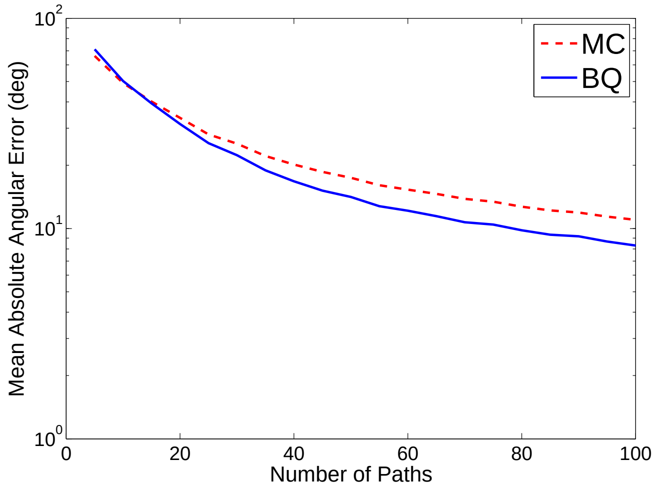

Figure 9. Results for the LQR using Model 2 without sparsification. Shown the mean absolute angular error of the MC and BQ gradient estimations as a function of the number of sample paths M.

tion

$$r_t = x_t^2 + 0.1a_t^2 + n_r \ ; \ n_r \sim \mathcal{N}(0, \sigma^2) \ ; \ \sigma^2 = 0.1$$

In Model 2, we model this by the measurement noise covariance matrix $\mathbf{\Sigma} = T\sigma^2\boldsymbol{I}$, where $T = 20$ is the path length in this problem. Since each reward r_t is a Gaussian random variable with variance σ^2, the return $R(\xi) = \sum_{t=0}^{T-1} r_t$ will also be a Gaussian random variable with variance $T\sigma^2$. Figures 10, 11, and 12 show the mean squared error, the mean and the variance of the gradient estimates, and the mean absolute angular error for the BQ (Model 2 without sparsification) and MC estimation methods for this problem.

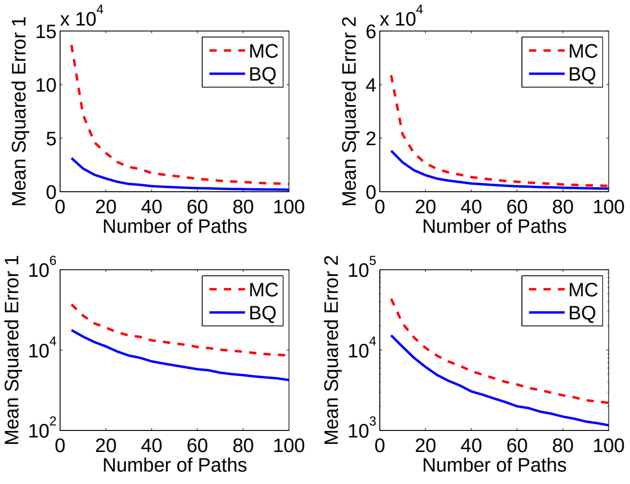

Figure 10. Results for the LQR when the reward function is corrupted by an i.i.d. Gaussian noise using Model 2 without sparsification. Shown are the MSE of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M in linear (top row) and logarithmic (bottom row) scales.

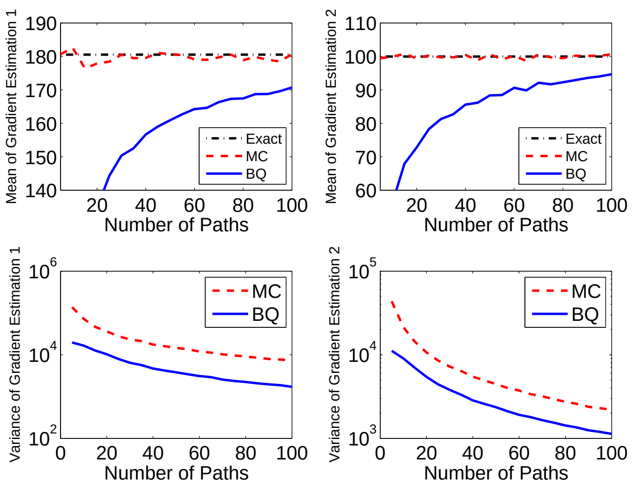

Figure 11. Results for the LQR when the reward function is corrupted by an i.i.d. Gaussian noise using Model 2 without sparsification. Shown are the mean and variance of the MC and BQ estimates of the first (left column) and second (right column) components of the gradient as a function of the number of sample paths M.

The experiments of this section indicate that the performance of the MC gradient estimate improves as $\frac{1}{M}$, where the performance of the BQ gradient estimate improves at a higher rate.

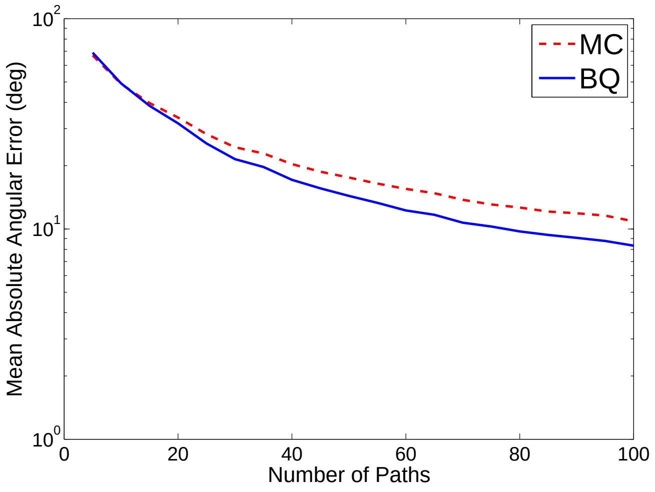

Figure 12. Results for the LQR when the reward function is corrupted by an i.i.d. Gaussian noise using Model 2 without sparsification. Shown is the mean absolute angular error of the MC and BQ gradient estimations as a function of the number of sample paths M.

5.2.2. Policy Optimization

In this section, we use Bayesian policy gradient (BPG) to train the parameters of the LQR problem. Figure 13 shows the performance of the BPG algorithm with the regular (BPG) and the natural (BPNG) gradient estimates, versus a MC-based policy gradient (MCPG) algorithm, for sample sizes $M = 5, 10$, and 40. M is the number of sample paths used by these methods for estimating the gradient of a policy. We use Algorithm 2 with the number of updates set to $N = 100$, and Model 1 without sparsification for the BPG and BPNG methods. Since Algorithm 2 computes the Fisher information matrix for each set of policy parameters, the estimate of the natural gradient is provided at little extra cost at each step. The returns obtained by these methods are averaged over 10^4 runs for sample sizes 5 and 10, and over 10^3 runs for sample size 40. The policy parameters are initialized randomly at each run. In order to ensure that the learning algorithms cannot exceed an acceptable parameter range, the policy parameters are defined as $c = -1.999 + 1.998/(1 + e^{\nu_1})$ and $\sigma = 0.001 + 1/(1 + e^{\nu_2})$. Thus, the policy parameter vector becomes $\boldsymbol{\theta} = (\nu_1\ ,\ \nu_2)^\top$. The optimal solution is $c^* \approx -0.92$ and $\sigma^* = 0.001$ $(\eta_B(c^*, \sigma^*) = 0.1003)$ corresponding to $\nu_1^* \approx -0.16$ and $\nu_2^* \to \infty$.

Table 2 summarizes the learning rates used in the experiments of this section. We use two different learn-

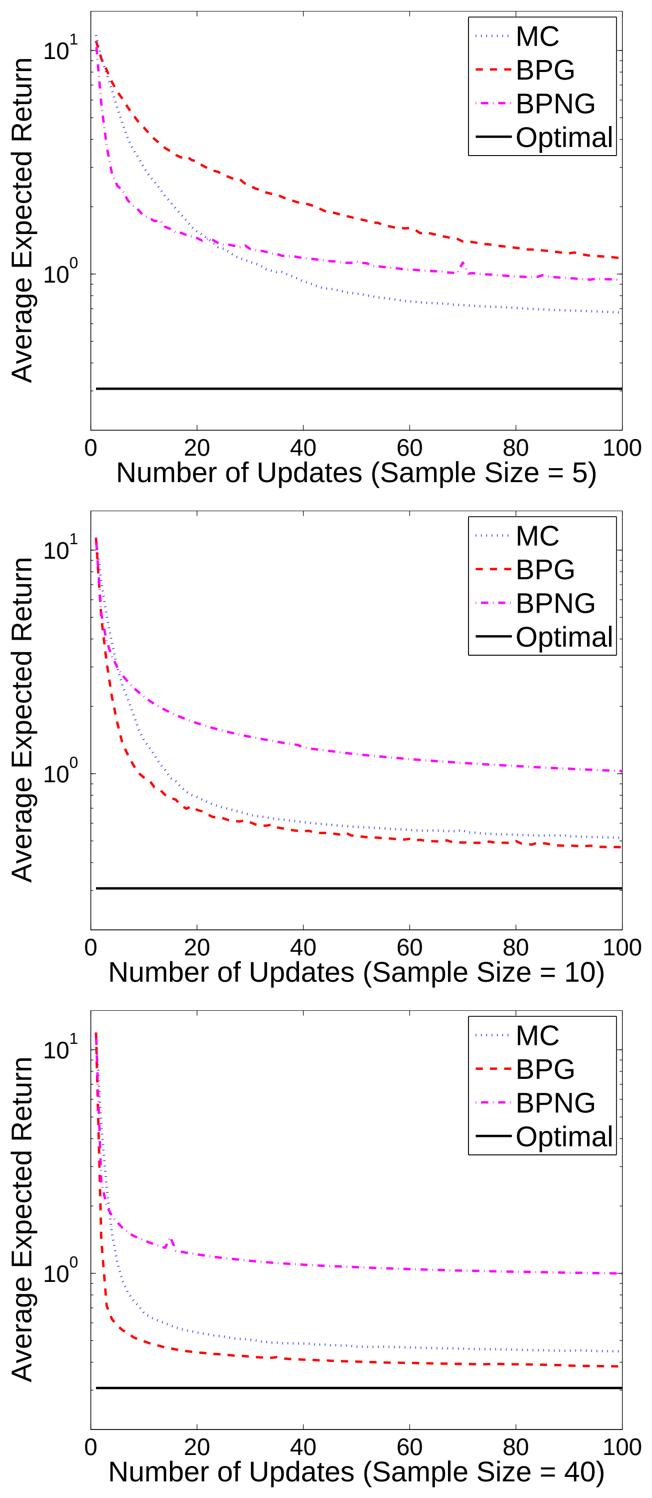

Figure 13. This figure compares the average expected returns of a Bayesian policy gradient algorithm using regular (BPG) and natural (BPNG) gradient estimates, with the average expected return of a MC-based policy gradient algorithm (MCPG) for three different sample sizes.

ing rates for the two components of the gradient. For a fixed sample size, each method uses a fixed learning rate vector, and the learning rates are not tuned during a trial. We tried many learning rates and those in Table 2 yielded the best performance. The selected learning rates for BPNG are significantly larger than those for BPG and MCPG as shown in Table 2. This explains why BPNG initially learns faster than BPG and MCPG, but eventually performs worse. We believe that this can be fixed by suitably reducing the learning rates during the execution of the algorithm. We leave this to future work.

	MCPG	BPG	BPNG
$M = 5$	$(0.01 \quad 0.05)$	$(0.01 \quad 0.07)$	$(0.05 \quad 0.5)$
$M = 10$	$(0.05 \quad 0.1)$	$(0.09 \quad 0.1)$	$(0.1 \quad 0.7)$
$M = 40$	$(0.1 \quad 0.15)$	$(0.1 \quad 0.4)$	$(0.8 \quad 0.9)$

Table 2. Learning rates used by the policy gradient algorithms in the experiments of this section.

Figure 13 shows that the MCPG algorithm performs better than the BPG algorithm only for very small sample size ($M = 5$). This phenomenon is also reported in Rasmussen and Ghahramani (2003). However, the experiments of Section 5.2.1 indicate that the BQ method performs better than the MC method even for very small sample size $M = 5$. Thus, we believe that using a suitable learning rate schedule, the BPG algorithm can perform better than the MCPG algorithm even for very small sample sizes.

6. Discussion

In this paper we proposed an alternative approach to conventional frequentist policy gradient estimation procedures, which is based on the Bayesian view. Our algorithms use GPs to define a prior distribution over the gradient of the expected return, and compute the posterior, conditioned on the observed data. The experimental results shown here are encouraging, but we conjecture that even higher gains may be attained using this approach. This calls for additional theoretical and empirical work.

Although the policy updating algorithm proposed here (Algorithm 2) uses only the posterior mean of the gradient in its updates, we hope that more elaborative algorithms can be devised that would make judicious use of the covariance information provided by the gradient estimation algorithm (Algorithm 1). Two obvious possibilities are: 1) risk-aware selection of the update step-size and direction, and 2) using the variance in a termination condition for Algorithm 1.

Other interesting directions include 1) investigating other possible partitions of the integrand in the expression for $\nabla\eta_B(\boldsymbol{\theta})$ into a GP term f and a known term p, 2) using other types of kernel functions, such as sequence kernels, 3) combining our approach with MDP model estimation, to allow transfer of learning between different policies, 4) investigating methods for learning the Fisher information matrix, 5) extending the Bayesian approach to Actor-Critic type of algorithms, possibly by combining Bayesian policy gradient with the Gaussian process temporal difference (GPTD) algorithms of Engel et al. (2003); Engel et al. (2005).

Acknowledgments

We would like to thank Rich Sutton and Dale Schuurmans for helpful discussions. M.G. would like to thank Shie Mannor for his useful comments at the early stages of this work. M.G. is supported by iCORE and Y.E. is partially supported by an Alberta Ingenuity fellowship.

References

Aleksandrov, V., Sysoyev, V., & Shemeneva, V. (1968). Stochastic optimization. *Engineering Cybernetics*, *5*, 11–16.

Bagnell, J., & Schneider, J. (2003). Covariant policy search. *Proceedings of the International Joint Conference on Artificial Intelligence.*

Baxter, J., & Bartlett, P. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Berger, J., & Wolpert, R. (1984). *The likelihood principle.* Institute of Mathematical Statistics, Hayward, CA.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming.* Athena Scientific.

Csató, L., & Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, *14*, 641–668.

Engel, Y., Mannor, S., & Meir, R. (2002). Sparse online greedy support vector regression. *Proceedings of the Thirteenth European Conference on Machine Learning* (pp. 84–96).

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets bellman: The Gaussian process approach to temporal difference learning. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 154–161).

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement learning with Gaussian processes. *Proceedings of the Twenty Second International Conference on Machine Learning* (pp. 201–208).

Glynn, P. (1986). Stochastic approximation for Monte Carlo optimization. *Proceedings of the Winter Simulation Conference* (pp. 356–365).

Glynn, P. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*, 75–84.

Glynn, P., & L'Ecuyer, P. (1995). Likelihood ratio gradient estimation for regenerative stochastic recursions. *Advances in Applied Probability*, *27*, 1019–1053.

Greensmith, E., Bartlett, P., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*, 1471–1530.

Jaakkola, T., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers. *Proceedings of Advances in Neural Information Processing Systems 11.* MIT Press.

Kakade, S. (2002). A natural policy gradient. *Proceedings of Advances in Neural Information Processing Systems 14.*

Kimura, H., Yamamura, M., & Kobayashi, S. (1995). Reinforcement learning by stochastic hill-climbing on discounted reward. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 295–303).

Marbach, P. (1998). *Simulated-based methods for Markov decision processes.* Doctoral dissertation, Massachusetts Institute of Technology.

O'Hagan, A. (1987). Monte Carlo is fundamentally unsound. *The Statistician*, *36*, 247–249.

O'Hagan, A. (1991). Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, *29*, 245–260.

Peters, J., Vijayakumar, S., & Schaal, S. (2003). Reinforcement learning for humanoid robotics. *Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots.*

Poincaré, H. (1896). *Calcul des probabilités.* Georges Carré, Paris.

Puterman, M. (1994). *Markov decision processes.* Wiley Interscience.

Rasmussen, C., & Ghahramani, Z. (2003). Bayesian Monte Carlo. *Proceedings of Advances in Neural Information Processing Systems 15* (pp. 489–496). MIT Press.

Reiman, M., & Weiss, A. (1986). Sensitivity analysis via likelihood ratios. *Proceedings of the Winter Simulation Conference.*

Reiman, M., & Weiss, A. (1989). Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, *37*.

Rubinstein, R. (1969). *Some problems in Monte Carlo optimization.* Doctoral dissertation, Polytechnic Institute, Riga, Latvia.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis.* Cambridge University Press.

Sutton, R., & Barto, A. (1998). *An introduction to reinforcement learning.* MIT Press.

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Proceedings of Advances in Neural Information Processing Systems 12* (pp. 1057–1063).

Weaver, L., & Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence* (pp. 538–545).

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.