Bayesian Actor-Critic Algorithms

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

Abstract

We[1] present a new actor-critic learning model in which a Bayesian class of non-parametric critics, using Gaussian process temporal difference learning is used. Such critics model the state-action value function as a Gaussian process, allowing Bayes' rule to be used in computing the posterior distribution over state-action value functions, conditioned on the observed data. Appropriate choices of the prior covariance (kernel) between state-action values and of the parametrization of the policy allow us to obtain closed-form expressions for the posterior distribution of the gradient of the average discounted return with respect to the policy parameters. The posterior mean, which serves as our estimate of the policy gradient, is used to update the policy, while the posterior covariance allows us to gauge the reliability of the update.

1. Introduction

Actor-Critic (AC) algorithms, first proposed by Barto et al. (1983), borrow elements from the two major families of Reinforcement learning (RL) algorithms. Borrowing from value-function based methods, a value function estimate is maintained (the critic). As in policy based methods, an actor maintains a separately parameterized stochastic action-selection policy. While the role of the actor is to select actions, the role of the critic is to evaluate the performance of the actor. This evaluation is used to provide the actor with a signal that allows it to improve its performance, typically by updating its parameters along an estimate of the gradient of some measure of performance, with respect to the actor's parameters. When the representations used for the actor and the critic are compatible, in a sense explained below, the resulting AC algorithm is simple, elegant and provably convergent to a local maximum of the performance measure used by the critic (under appropriate conditions) (Sutton et al., 2000; Konda & Tsitsiklis, 2000).

Current AC algorithms are based on parametric critics that are updated to optimize frequentist fitness criteria. For instance, TD(λ) - a commonly used critic, uses a linear combination of fixed basis functions to approximate the value function. In this case the critic's parameters are the linear combination coefficients[2] (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996). By "frequentist" we mean algorithms that return a point estimate of the value function, rather than a complete posterior distribution computed using Bayes' rule. TD(1), for instance, may be shown to return a maximum likelihood (ML) estimate of the value function, based on a linear-normal statistical model (Engel, 2005).

In Sutton et al. (2000) and Konda and Tsitsiklis (2000) it was shown that if the policy belongs to an exponential family, that is $\mu(\boldsymbol{a}, \mathbf{x}; \boldsymbol{\theta}) = \exp\left(\sum_{i=1}^{n} \theta_i \phi_i(\mathbf{x}, \boldsymbol{a})\right) / Z(\mathbf{x}; \boldsymbol{\theta})$, it is only necessary that the critic learns an estimate of the projection of the (state-action) value function on the span of $\{\phi_i(\cdot)\}_{i=1}^{n}$, rather than an estimate of the value function itself, without introducing bias in the gradient estimate. Notwithstanding, it is noted in Konda and Tsitsiklis (2000) that in practice it may prove advantageous to allow the critic to search in a higher dimensional space that contains span $(\{\phi_i(\cdot)\}_{i=1}^{n})$ as a proper subset. This may help in reducing the variance of the gradient estimate, as well as improving the stability of the resulting algorithm.

A Bayesian class of critics, based on Gaussian processes (GPs) has been recently proposed by Engel et al.

[1] Both authors contributed equally to this work.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

[2] TD(λ) may be used with other function approximation schemes, however its convergence is not guaranteed then.

(2003); Engel et al. (2005); Engel (2005). By their Bayesian nature, these algorithms return a full posterior distribution over value functions. Moreover, while these algorithms may be used to learn a parametric representation for the posterior, they are generally capable of searching for value functions in an infinite-dimensional Hilbert space of functions, resulting in a non-parametric posterior.

In this paper we propose an AC algorithm that incorporates a GP temporal-difference (GPTD) algorithm as its critic. However, rather than merely plugging-in our critic into an existing AC algorithm, we show how the posterior moments returned by the GPTD critic allow us to obtain closed-form expressions for the posterior moments of the policy gradient. This is made possible by utilizing the Fisher kernel as our prior covariance kernel for the GPTD state-action *advantage* values (Shawe-Taylor & Cristianini, 2004). This is a natural extension of the *Bayesian policy gradient* (BPG) approach proposed in (Ghavamzadeh & Engel, 2007). However, in BPG the basic observable unit, upon which learning and inference are based, is a complete trajectory. The Bayesian AC (BAC) approach proposed here takes advantage of the Markov property of the system trajectories and uses individual state-action-reward transitions as its basic observable unit. This helps reduce variance in the gradient estimates, resulting in steeper learning curves when compared to BPG and the classic Monte-Carlo approach.

2. Preliminaries

Reinforcement Learning (RL) (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998) is a class of learning problems in which an agent interacts with an unfamiliar, dynamic and stochastic environment, where the agent's goal is to optimize some measure of its long-term performance. This interaction is conventionally modeled as a Markov decision process (MDP). Let $\mathcal{P}(\mathcal{S})$ be the set of probability distributions on (Borel) subsets of a set $\mathcal{S}$. 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; $q(\cdot|\boldsymbol{a}, \mathbf{x}) \in \mathcal{P}(\mathbb{R})$ is the probability distribution over rewards; $P(\cdot|\boldsymbol{a}, \mathbf{x}) \in \mathcal{P}(\mathcal{X})$ is the transition probability distribution; (we assume that P and q are stationary); and $P_0(\cdot) \in \mathcal{P}(\mathcal{X})$ is the initial state distribution. We denote the random variable distributed according to $q(\cdot|\boldsymbol{a}, \mathbf{x})$ by $R(\mathbf{x}, \boldsymbol{a})$. 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* $\mu(\cdot|\mathbf{x}) \in \mathcal{P}(\mathcal{A})$ is a probability distribution over actions, conditioned on the current state. The MDP controlled by the policy μ induces a Markov chain over state-action pairs $\mathbf{z}_t = (\mathbf{x}_t, \boldsymbol{a}_t) \in \mathcal{Z} = \mathcal{X} \times \mathcal{A}$, with a transition probability density $P^\mu(\mathbf{z}_t|\mathbf{z}_{t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{a}_{t-1})\mu(\boldsymbol{a}_t|\mathbf{x}_t)$, and an initial state density $P_0^\mu(\mathbf{z}_0) = P_0(\mathbf{x}_0)\mu(\boldsymbol{a}_0|\mathbf{x}_0)$. We generically denote by $\boldsymbol{\xi} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T) \in \Xi$, $T \in \{0, 1, \ldots, \infty\}$, a path generated by this Markov chain. The probability (density) of such a path is given by

$$\Pr(\boldsymbol{\xi}|\mu) = P_0^\mu(\mathbf{z}_0) \prod_{t=1}^{T} P^\mu(\mathbf{z}_t|\mathbf{z}_{t-1}). \tag{1}$$

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

$$\eta(\mu) = \mathbf{E}(\rho(\boldsymbol{\xi})) = \int_\Xi \bar{\rho}(\boldsymbol{\xi}) \Pr(\boldsymbol{\xi}|\mu) d\boldsymbol{\xi}. \tag{2}$$

Let us define the t-step state-action occupancy density and the state-action value function, respectively:

$$P_t^\mu(\mathbf{z}_t) = \int_{\mathcal{Z}^t} d\mathbf{z}_0 \ldots d\mathbf{z}_{t-1} P_0^\mu(\mathbf{z}_0) \prod_{i=1}^{t} P^\mu(\mathbf{z}_i|\mathbf{z}_{i-1}),$$

$$Q(\mathbf{z}) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{z}_t)|\mathbf{z}_0 = \mathbf{z}\right].$$

It can be shown that under certain regularity conditions (Sutton et al., 2000),

$$\eta(\mu) = \int_{\mathcal{Z}} d\mathbf{z} \pi^\mu(\mathbf{z}) \bar{R}(\mathbf{z}), \tag{3}$$

where $\bar{R}(\mathbf{z})$ is the mean reward for the state-action pair $\mathbf{z}$, and

$$\pi^\mu(\mathbf{z}) = \sum_{i=0}^{\infty} \gamma^i P_i^\mu(\mathbf{z})$$

is a discounted weighting of state-action pairs. Integrating $\boldsymbol{a}$ out of $\pi^\mu(\mathbf{x}, \boldsymbol{a})$ results in the corresponding discounted weighting of states $\pi(\mathbf{x}) = \int_{\mathcal{A}} d\boldsymbol{a} \pi^\mu(\mathbf{x}, \boldsymbol{a})$.

In AC methods, one defines a class of smoothly parameterized stochastic policies $\{\mu(\cdot|\mathbf{x}; \boldsymbol{\theta}), \mathbf{x} \in \mathcal{X}, \boldsymbol{\theta} \in \Theta\}$. Algorithms typically estimate the gradient of the expected return (2) with respect to the policy parameters $\boldsymbol{\theta}$ from observed system trajectories, and then improve the policy by adjusting its parameters in the direction of the gradient. The policy gradient theorem (Marbach, 1998, Prop. 1, Sutton et al., 2000, Thm. 1,

Konda & Tsitsiklis, 2000, Thm. 1) states that the gradient of the expected return $\eta(\boldsymbol{\theta}) = \eta(\mu(\cdot|\cdot;\boldsymbol{\theta}))$ is given by

$$\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta}) = \int_{\mathcal{Z}} d\mathbf{x}d\boldsymbol{a}\pi(\mathbf{x})\nabla_{\boldsymbol{\theta}}\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})Q(\mathbf{x},\boldsymbol{a}) \quad (4)$$
$$= \int_{\mathcal{Z}} d\mathbf{z}\pi^{\mu}(\mathbf{z})\nabla_{\boldsymbol{\theta}}\log\left(\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})\right)Q(\mathbf{z}).$$

Moreover, by making two additional assumptions we may replace the exact (but unknown) state-action value function $Q(\mathbf{z})$ in 4 by an approximate state-action value function $\hat{Q}(\mathbf{z})$ (Sutton et al., 2000, Thm. 2, Konda & Tsitsiklis, 2000):

Assumption 1 (Compatibility) *Suppose that $\hat{Q}(\mathbf{z})$ is parametrized by a vector $\mathbf{w}$ of n parameters, i.e., $\hat{Q}(\mathbf{z}) = \hat{Q}(\mathbf{z};\mathbf{w})$, then $\nabla_{\mathbf{w}}\hat{Q}(\mathbf{x},\boldsymbol{a};\mathbf{w}) = \nabla_{\boldsymbol{\theta}}\log(\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta}))$.*

Assumption 2 (Projection) *$\hat{Q}(\mathbf{z};\mathbf{w})$ is the projection of $Q(\mathbf{z})$ onto the space $\{\hat{Q}(\mathbf{z};\mathbf{w})|\mathbf{w}\in\mathbb{R}^n\}$, with respect to a weighted L^2-norm weighted by $\pi^{\mu}(\mathbf{z})$. In other words, $\mathbf{w}$ minimizes $\int_{\mathcal{Z}} d\mathbf{z}\pi^{\mu}(\mathbf{z})(\hat{Q}(\mathbf{z};\mathbf{w}) - Q(\mathbf{z}))^2$.*

Under these assumptions we have

$$\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta}) = \int_{\mathcal{Z}} d\mathbf{z}\pi^{\mu}(\mathbf{z})\nabla\log\left(\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})\right)\hat{Q}(\mathbf{z};\mathbf{w}). \quad (5)$$

A convenient choice that ensures compatibility between the policy and the state-action value representations is $\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta}) = \exp\left(\sum_{i=1}^{n}\boldsymbol{\phi}(\mathbf{z})^{\top}\boldsymbol{\theta}\right)/Z(\mathbf{x};\boldsymbol{\theta})$, and $\hat{Q}(\mathbf{z};\mathbf{w}) = \mathbf{w}^{\top}\left(\boldsymbol{\phi}(\mathbf{z}) - \mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\boldsymbol{\phi}(\mathbf{z})\right) + b(\mathbf{x})$, where $\mathbf{E}_{\boldsymbol{a}|\mathbf{x}}[\cdot] = \int_{\mathcal{A}} d\boldsymbol{a}\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})[\cdot]$, and $b : \mathcal{X}\rightarrow\mathbb{R}$ an arbitrary function that does not depend on $\mathbf{w}$. Note that $\mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\hat{Q}(\mathbf{z};\mathbf{w}) = b(\mathbf{x})$, since $\mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\left(\boldsymbol{\phi}(\mathbf{z}) - \mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\boldsymbol{\phi}(\mathbf{z})\right) = 0$. This means that if $\hat{Q}(\mathbf{z};\mathbf{w})$ approximates $Q(\mathbf{z})$ then $b(\mathbf{x})$ must approximate the value function $V(\mathbf{x})$. The term $\hat{A}(\mathbf{z};\mathbf{w}) = \hat{Q}(\mathbf{z};\mathbf{w}) - b(\mathbf{x}) = \mathbf{w}^{\top}\left(\boldsymbol{\phi}(\mathbf{z}) - \mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\boldsymbol{\phi}(\mathbf{z})\right)$ approximates the *advantage function* $A(\mathbf{z}) = Q(\mathbf{z}) - V(\mathbf{x})$ (Baird, 1993).

3. Bayesian Actor-Critic

In Ghavamzadeh and Engel (2007) a class of Bayesian policy gradient algorithms is proposed. In that paper the gradient of the expected return is expressed as

$$\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta}) = \int \rho(\boldsymbol{\xi})\frac{\nabla\Pr(\boldsymbol{\xi};\boldsymbol{\theta})}{\Pr(\boldsymbol{\xi};\boldsymbol{\theta})}\Pr(\boldsymbol{\xi};\boldsymbol{\theta})d\boldsymbol{\xi}, \quad (6)$$

where samples of $\rho(\boldsymbol{\xi})$ are obtained by observing complete trajectories from initial state to terminal state in an episodic setting. The term $\frac{\nabla\Pr(\boldsymbol{\xi};\boldsymbol{\theta})}{\Pr(\boldsymbol{\xi};\boldsymbol{\theta})} = \sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}}\log\mu(\boldsymbol{a}_t|\mathbf{x}_t;\boldsymbol{\theta})$ may be computed using only the (known) policy μ. In one of the Bayesian policy gradient models proposed there (Model 2), the $\rho(\boldsymbol{\xi})$ term is modeled as a Gaussian process. Its posterior moments are then evaluated by conditioning on the observed returns of complete trajectories, essentially amounting to GP regression on these returns. Since the transformation from $\rho(\boldsymbol{\xi})$ to $\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})$ is performed by a linear integral operator, the posterior distribution of $\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})$ is also Gaussian, and for an appropriate choice of a prior covariance kernel $k(\boldsymbol{\xi},\boldsymbol{\xi}') = \mathbf{Cov}[\rho(\boldsymbol{\xi}),\rho(\boldsymbol{\xi}')]$, closed-form expressions for the posterior moments are obtained. It should be noted that, since the models and algorithms of Ghavamzadeh and Engel (2007) consider complete trajectories as the basic observable unit, they do not require the dynamics within each trajectory to be of any specific form. In particular, it is not necessary for the dynamics to have the Markov property, allowing the resulting algorithms to handle partially observable MDPs, Markov games and other non-Markovian systems. On the down side, these algorithms can not take advantage of the Markov property in systems that have this property.

In this paper we start instead with the expression for the policy gradient given in Eq. 4. We will place a GP prior over state-action value functions using a prior covariance kernel defined on state-action pairs $k(\mathbf{z},\mathbf{z}') = \mathbf{Cov}[Q(\mathbf{z}),Q(\mathbf{z}')]$. We will then compute the GP posterior, conditioned on the sequence of individual observed transitions. Again, by an appropriate choice of kernel we will be able to get closed-form expressions to the posterior moments of $\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})$. Fortunately, well developed machinery for computing the posterior moments of $Q(\mathbf{x},\boldsymbol{a})$ is provided in a series of papers by Engel et al. (2003); Engel et al. (2005) (for a thorough treatment see Engel, 2005).

Let us briefly review some of the main results pertaining to the Gaussian process temporal difference (GPTD) model proposed in Engel et al. (2005). The GPTD model is based on a statistical generative model relating the observed reward signal R to the unobserved state-action value function Q.

$$R(\mathbf{z}_i) = Q(\mathbf{z}_i) - \gamma Q(\mathbf{z}_{i+1}) + N(\mathbf{z}_i,\mathbf{z}_{i+1}). \quad (7)$$

$N(\mathbf{z}_i,\mathbf{z}_{i+1})$ is a zero-mean noise signal that accounts for the discrepancy between $R(\mathbf{z}_i)$ and $Q(\mathbf{z}_i) - \gamma Q(\mathbf{z}_{i+1})$. Let us define the finite-dimensional processes R_t, Q_t, N_t and the $t\times(t+1)$ matrix $\mathbf{H}_t$:

$$R_t = \left(R(\mathbf{z}_0),\ldots,R(\mathbf{z}_t)\right)^{\top}, Q_t = \left(Q(\mathbf{z}_0),\ldots,Q(\mathbf{z}_t)\right)^{\top},$$
$$N_t = \left(N(\mathbf{z}_0,\mathbf{z}_1),\ldots,N(\mathbf{z}_{t-1},\mathbf{z}_t)\right)^{\top}, \quad (8)$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & -\gamma \end{bmatrix}. \quad (9)$$

The set of equations (7) for $i = 0, \dots, t$ may be written as

$$R_{t-1} = \mathbf{H}_t Q_t + N_t. \quad (10)$$

Under certain assumptions on the distribution of the discounted return random process (Engel et al., 2005), the covariance of the noise vector N_t is given by

$$\begin{aligned} \mathbf{\Sigma}_t &= \sigma^2 \mathbf{H}_t \mathbf{H}_t^\top \quad (11) \\ &= \sigma^2 \begin{bmatrix} 1+\gamma^2 & -\gamma & 0 & \dots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \dots & -\gamma & 1+\gamma^2 \end{bmatrix}. \end{aligned}$$

In episodic tasks, if $\mathbf{z}_{t-1}$ is the last state-action pair in the episode (that is, $\mathbf{x}_t$ is zero-reward absorbing terminal state), $\mathbf{H}_t$ becomes a square $t \times t$ invertible matrix of the form (9) with its last column removed. The effect on the noise covariance matrix $\mathbf{\Sigma}_t$ is that the bottom-right element becomes 1 instead of $1+\gamma^2$.

Placing a Gaussian process prior on Q and assuming that N_t is also normally distributed, we may use Bayes' rule to obtain the posterior moments of Q:

$$\begin{aligned} \hat{Q}_t(\mathbf{z}) &= \mathbf{E}\left[Q(\mathbf{z}) | \mathcal{D}_t\right] = \mathbf{k}_t(\mathbf{z})^\top \boldsymbol{\alpha}_t, \quad (12) \\ \hat{S}_t(\mathbf{z}, \mathbf{z}') &= \mathbf{Cov}\left[Q(\mathbf{z}), Q(\mathbf{z}') | \mathcal{D}_t\right] \\ &= k(\mathbf{z}, \mathbf{z}') - \mathbf{k}_t(\mathbf{z})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{z}'), \end{aligned}$$

where $\mathcal{D}_t$ denotes the observed data until and including time t. We used here the following definitions:

$$\begin{aligned} \mathbf{k}_t(\mathbf{z}) &= \left(k(\mathbf{z}_0, \mathbf{z}), \dots, k(\mathbf{z}_t, \mathbf{z})\right)^\top, \\ \mathbf{K}_t &= \left[\mathbf{k}_t(\mathbf{z}_0), \mathbf{k}_t(\mathbf{z}_1), \dots, \mathbf{k}_t(\mathbf{z}_t)\right], \\ \boldsymbol{\alpha}_t &= \mathbf{H}_t^\top \left(\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t\right)^{-1} R_{t-1}, \\ \mathbf{C}_t &= \mathbf{H}_t^\top \left(\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t\right)^{-1} \mathbf{H}_t. \quad (13) \end{aligned}$$

We are now in a position to describe the main idea behind our BAC approach. Making use of the linearity of Eq. 4 in Q, and denoting $\boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) = \pi^\mu(\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log\left(\mu(\boldsymbol{a}|\mathbf{x}; \boldsymbol{\theta})\right)$, we obtain the following expressions for the posterior moments of the policy gradient (O'Hagan, 1991):

$$\begin{aligned} \mathbf{E}[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t] &= \int_{\mathcal{Z}} d\mathbf{z} \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) \hat{Q}_t(\mathbf{z}), \quad (14) \\ \mathbf{Cov}\left[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t\right] &= \int_{\mathcal{Z}^2} d\mathbf{z} d\mathbf{z}' \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) \hat{S}_t(\mathbf{z}, \mathbf{z}') \boldsymbol{g}(\mathbf{z}'; \boldsymbol{\theta})^\top. \end{aligned}$$

Substituting the expressions for the posterior moments (12) into Eq. 14, we get

$$\begin{aligned} \mathbf{E}[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t] &= \int_{\mathcal{Z}} d\mathbf{z} \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) \mathbf{k}_t(\mathbf{z})^\top \boldsymbol{\alpha}_t, \quad (15) \\ \mathbf{Cov}\left[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t\right] &= \\ \int_{\mathcal{Z}^2} d\mathbf{z} d\mathbf{z}' \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) &\left(k(\mathbf{z}, \mathbf{z}') - \mathbf{k}_t(\mathbf{z})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{z}')\right) \boldsymbol{g}(\mathbf{z}'; \boldsymbol{\theta})^\top. \end{aligned}$$

The equations above provide us with the general form of the posterior policy gradient moments. We are now left with a computational issue; namely, how to compute the integrals appearing in these expressions? We need to be able to evaluate the following integrals:

$$\begin{aligned} \mathbf{U}_t &= \int_{\mathcal{Z}} d\mathbf{z} \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) \mathbf{k}_t(\mathbf{z})^\top, \quad (16) \\ \mathbf{V} &= \int_{\mathcal{Z}^2} d\mathbf{z} d\mathbf{z}' \boldsymbol{g}(\mathbf{z}; \boldsymbol{\theta}) k(\mathbf{z}, \mathbf{z}') \boldsymbol{g}(\mathbf{z}'; \boldsymbol{\theta})^\top. \end{aligned}$$

Using these definitions we may write the gradient posterior moments compactly as

$$\begin{aligned} \mathbf{E}[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t] &= \mathbf{U}_t \boldsymbol{\alpha}_t, \quad (17) \\ \mathbf{Cov}\left[\nabla_{\boldsymbol{\theta}} \eta | \mathcal{D}_t\right] &= \mathbf{V} - \mathbf{U}_t \mathbf{C}_t \mathbf{U}_t^\top. \end{aligned}$$

In order to render these integrals analytically tractable we choose our prior covariance kernel to be the sum of an arbitrary state-kernel k_x and the (invariant) Fisher kernel k_F between state-action pairs (Shawe-Taylor & Cristianini, 2004, Chap. 12). More specifically, let us denote the *score* vector and the *Fisher information* matrix corresponding to the policy $\mu(\cdot|\cdot; \boldsymbol{\theta})$, respectively by

$$\begin{aligned} \mathbf{u}(\mathbf{z}) &= \nabla_{\boldsymbol{\theta}} \log(\mu(\boldsymbol{a}|\mathbf{x}; \boldsymbol{\theta})), \quad (18) \\ \mathbf{G} &= \mathbf{E}_{\mathbf{z}}\left[\mathbf{u}(\mathbf{z}) \mathbf{u}(\mathbf{z})^\top\right] = \int_{\mathcal{Z}} d\mathbf{z} \pi^\mu(\mathbf{z}) \mathbf{u}(\mathbf{z}) \mathbf{u}(\mathbf{z})^\top. \end{aligned}$$

It is readily verified that $\mathbf{E}_{\boldsymbol{a}|\mathbf{x}} \mathbf{u}(\mathbf{z}) = \mathbf{0}$.

The (policy dependent) Fisher information kernel and our overall state-action kernel are then given by

$$\begin{aligned} k_F(\mathbf{z}, \mathbf{z}') &= \mathbf{u}(\mathbf{z})^\top \mathbf{G}^{-1} \mathbf{u}(\mathbf{z}'), \quad (19) \\ k(\mathbf{z}, \mathbf{z}') &= k_x(\mathbf{x}, \mathbf{x}') + k_F(\mathbf{z}, \mathbf{z}'), \end{aligned}$$

respectively. A nice property of the Fisher kernel is that while $k_F(\mathbf{z}, \mathbf{z}')$ depends on the policy, it is invariant to policy reparameterization. In other words, it only depends on the actual probability mass assigned to each action, and not on its explicit dependence on the policy parameters.

As mentioned above, another attractive property of this particular choice of kernel is that it renders the integrals in (16) analytically tractable. It is a matter of basic (but tedious) algebra to prove the following:

Proposition 1 *Let $k(\mathbf{z},\mathbf{z}') = k_x(\mathbf{x},\mathbf{x}') + k_F(\mathbf{z},\mathbf{z}')$ for all $(\mathbf{z},\mathbf{z}') \in \mathcal{Z}^2$, where $k_x : \mathcal{X}^2 \to \mathbb{R}$ is an arbitrary positive definite kernel function. Then $\mathbf{U}_t$ and $\mathbf{V}$ from Eq. 16 satisfy*

$$\mathbf{U}_t = [\mathbf{u}(\mathbf{z}_0), \mathbf{u}(\mathbf{z}_1), \ldots, \mathbf{u}(\mathbf{z}_t)], \text{ and } \mathbf{V} = \mathbf{G}. \quad (20)$$

An immediate consequence of Proposition 1 is that, in order to compute the posterior moments of the policy gradient, we only need to be able to evaluate (or estimate) the score vectors $\mathbf{u}(\mathbf{z}_i)$ and the Fisher information matrix $\mathbf{G}$ of our policy.

A convenient, as well as rather flexible choice for a space of policies to conduct our search in is a parametric exponential family. Namely,

$$\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta}) = \frac{1}{Z_{\boldsymbol{\theta}}(\mathbf{x})} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x},\boldsymbol{a})\right), \quad (21)$$

where $Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int_{\mathcal{A}} d\boldsymbol{a} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x},\boldsymbol{a})\right)$ is a normalizing factor, referred to as the *partition function*. It is easy to show that

$$\mathbf{u}(\mathbf{z}) = \boldsymbol{\phi}(\mathbf{x},\boldsymbol{a}) - \mathbf{E}_{\boldsymbol{a}|\mathbf{x}}\left[\boldsymbol{\phi}(\mathbf{x},\boldsymbol{a})\right]. \quad (22)$$

We are now left with the problem of evaluating $\mathbf{u}(\mathbf{z})$ and $\mathbf{G}$. For MDPs with a finite action space $\mathcal{A}$, the expectation $\mathbf{E}_{\boldsymbol{a}|\mathbf{x}}$ degenerates into a sum, which may be performed in time linear in the number of actions. If $|\mathcal{A}|$ is infinite, we may still be able to compute the expectation in certain cases. For instance, if $\mathcal{A} = \mathbb{R}$ and the policy features are $\boldsymbol{\phi}(\mathbf{z}) = (s_1(\mathbf{x})\boldsymbol{a}, s_2(\mathbf{x})\boldsymbol{a}^2)^\top$, then $\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})$ is a univariate normal probability density with state-dependent mean $m(\mathbf{x}) = -s_1(\mathbf{x})/(2s_2(\mathbf{x}))$ and variance $\sigma^2(\mathbf{x}) = -1/(2s_2(\mathbf{x}))$. The score vector is therefore $\mathbf{u} = \left(\boldsymbol{a} - m(\mathbf{x}), \boldsymbol{a}^2 - m(\mathbf{x})^2 - \sigma^2(\mathbf{x})\right)^\top$.

Evaluating the Fisher information matrix $\mathbf{G}$ is somewhat more challenging, since on top of taking the expectation w.r.t. the policy $\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta})$, computing $\mathbf{G}$ involves an additional expectation over the state-occupancy density $\pi(\mathbf{x})$, which is not generally known. In most practical situations we therefore have to resort to estimating $\mathbf{G}$ from data. One straightforward method is to estimate $\mathbf{G}$ from a trajectory $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_t$ using the (unbiased) estimator (see Proposition 1 for the definition of $\mathbf{U}_t$):

$$\hat{\mathbf{G}}_t = \frac{1}{t+1}\sum_{i=0}^{t} \mathbf{u}(\mathbf{z}_i)\mathbf{u}(\mathbf{z}_i)^\top = \frac{1}{t+1}\mathbf{U}_t\mathbf{U}_t^\top. \quad (23)$$

Alg. 1 is a pseudocode sketch of the Bayesian actor-critic algorithm, using either the regular or the natural gradient in the policy update, and with $\mathbf{G}$ estimated using $\hat{\mathbf{G}}_t$.

Algorithm 1 Bayesian Actor-Critic

BAC$(\boldsymbol{\theta}, M, \varepsilon)$
- $\boldsymbol{\theta}$ initial policy parameters
- $M > 0$ episodes for gradient evaluation
- $\varepsilon > 0$ termination threshold

done = false
while not done **do**
  Run **GPTD** for M episodes. **GPTD** returns $\boldsymbol{\alpha}_t, \mathbf{C}_t, \mathbf{U}_t, \hat{\mathbf{G}}_t$ (13, 20, 23)
  $\Delta\boldsymbol{\theta} = \mathbf{U}_t\boldsymbol{\alpha}_t$ (regular gradient) or $\Delta\boldsymbol{\theta} = \hat{\mathbf{G}}_t^{-1}\mathbf{U}_t\boldsymbol{\alpha}_t$ (natural gradient)
  $\boldsymbol{\theta} := \boldsymbol{\theta} + \beta\Delta\boldsymbol{\theta}$
  if $|\Delta\boldsymbol{\theta}| < \varepsilon$ **then** done = true
end while
return $\boldsymbol{\theta}$

4. Theory

We can glean some useful insights into the BAC algorithm by representing our statistical model in a different, but equivalent form. The idea is to define a hybrid prior for the state-action GP, $Q = V + A$. The prior over value functions V will be non-parametric, while the prior over advantage functions A will be parametric. Denote by $\mathcal{GP}(\mathcal{S})$ the set of GPs indexed by the set $\mathcal{S}$. Let $\boldsymbol{\psi}(\mathbf{z}) = \mathbf{u}(\mathbf{z}) = \nabla_{\boldsymbol{\theta}} \log(\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta}))$, set $Q(\mathbf{z}) = V(\mathbf{x}) + A(\mathbf{z}) = V(\mathbf{x}) + W^\top\boldsymbol{\psi}(\mathbf{z})$, where W is a random vector with a prior distribution $W \sim \mathcal{N}(\mathbf{0}, \mathbf{G}^{-1})$. Let $V \in \mathcal{GP}(\mathcal{X})$ be a GP with prior mean $\mathbf{E}[V(\mathbf{x})] = 0$ and prior covariance $\mathbf{Cov}[V(\mathbf{x}), V(\mathbf{x}')] = k_x(\mathbf{x},\mathbf{x}')$, and further let W and V be a priori uncorrelated (and therefore independent), i.e., $\mathbf{Cov}[W, V(\mathbf{x})] = \mathbf{0}$ for all $\mathbf{x} \in \mathcal{X}$. Then, $Q \in \mathcal{GP}(\mathcal{Z})$ is a GP with a prior mean $\mathbf{E}[Q(\mathbf{z})] = 0$ and a prior covariance $k(\mathbf{z},\mathbf{z}') = \mathbf{Cov}\left[Q(\mathbf{z}), Q(\mathbf{z}')\right] = \mathbf{Cov}[V(\mathbf{x}), V(\mathbf{x}')] + \boldsymbol{\psi}(\mathbf{z})^\top\mathbf{E}\left[WW^\top\right]\boldsymbol{\psi}(\mathbf{z}') = k_x(\mathbf{x},\mathbf{x}') + \mathbf{u}(\mathbf{z})^\top\mathbf{G}^{-1}\mathbf{u}(\mathbf{z}') = k_x(\mathbf{x},\mathbf{x}') + k_F(\mathbf{z},\mathbf{z}')$. This alternative view of our original model leads to the following result.

Proposition 2 *Let the conditions in Proposition 1 hold, and let $Q(\mathbf{z}) = V(\mathbf{x}) + W^\top\mathbf{u}(\mathbf{z})$, then Assumption 1 holds.*

Proof For any instantiation $\mathbf{w}$ of W we have $\nabla_{\mathbf{w}} Q(\mathbf{z};\mathbf{w}) = \mathbf{u}(\mathbf{z}) = \nabla_{\boldsymbol{\theta}} \log(\mu(\boldsymbol{a}|\mathbf{x};\boldsymbol{\theta}))$.

Let us review the GPTD statistical model in this hybrid representation. The model equations are given by Eq. 7. Using the definitions in (8, 9), the normality of the noise vector N_t and the noise covariance from (11) we may write the joint (normal) distribution of W and

$V(\mathbf{x})$, conditioned on R_{t-1}:

$$\begin{pmatrix} W \\ V(\mathbf{x}) \end{pmatrix} \Big| R_{t-1} \sim \mathcal{N}\left\{ \begin{pmatrix} \hat{W}_t \\ \hat{V}_t(\mathbf{x}) \end{pmatrix}, \begin{bmatrix} \mathbf{S}_w, & \mathbf{s}_{wv} \\ \mathbf{s}_{wv}^\top, & s_v \end{bmatrix} \right\},$$

where, using the definitions in (13),

$$\begin{aligned} \hat{W}_t &= \mathbf{G}^{-1}\mathbf{U}_t\boldsymbol{\alpha}_t, \quad \hat{V}_t(\mathbf{x}) = \mathbf{k}_t^x(\mathbf{x})\boldsymbol{\alpha}_t, \\ \mathbf{S}_w &= \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{U}_t\mathbf{C}_t\mathbf{U}_t\mathbf{G}^{-1}, \\ \mathbf{s}_{wv}(\mathbf{x}) &= -\mathbf{G}^{-1}\mathbf{U}_t\mathbf{C}_t\mathbf{k}_t^x(\mathbf{x}), \\ s_v(\mathbf{x}) &= k_x(\mathbf{x},\mathbf{x}) - \mathbf{k}_t^x(\mathbf{x})^\top\mathbf{C}_t\mathbf{k}_t^x(\mathbf{x}), \end{aligned}$$

and $\mathbf{K}_t = \mathbf{K}_t^x + \mathbf{U}_t^\top\mathbf{G}^{-1}\mathbf{U}_t$ (used in $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$).

First, we observe that we are getting separate estimates for the value function ($\hat{V}_t(\mathbf{x})$) and the advantage function ($\hat{A}_t(\mathbf{z}) = \hat{W}_t^\top\mathbf{u}(\mathbf{z})$), the sum of which is an estimate of the state-action value function $\hat{Q}_t(\mathbf{x}) = \hat{V}_t(\mathbf{x}) + \hat{W}_t^\top\mathbf{u}(\mathbf{z})$. We are also getting confidence measures on these estimates via the posterior variance $s_v(\mathbf{x})$ of $V(\mathbf{x})$ and the posterior variance $\mathbf{u}(\mathbf{z})^\top\mathbf{S}_w\mathbf{u}(\mathbf{z})$ of $A(\mathbf{z})$. Also note that, although V and A are a priori uncorrelated, they do not remain so in the posterior, as $\mathbf{s}_{wv}(\mathbf{x})$ is not generally zero. Another interesting observation is that $\hat{W}_t$ is the posterior mean of the natural gradient of $\eta(\boldsymbol{\theta})$ – an analogous result to Thm. 1 in Kakade (2002).

It is instructive to investigate how the form of the posterior moments is influenced by making the approximation of replacing $\mathbf{G}$ with $\hat{\mathbf{G}}_t$. Let us start with the posterior mean (see Eq. 13 and 17).

$$\begin{aligned} &\mathbf{E}\left[\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})|\mathcal{D}_t\right] = \mathbf{U}_t\boldsymbol{\alpha}_t \\ =&\mathbf{U}_t\mathbf{H}_t^\top\left(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}R_{t-1} \\ =&\mathbf{U}_t\left(\mathbf{H}_t^\top\boldsymbol{\Sigma}_t^{-1}\mathbf{H}_t\mathbf{K}_t + \mathbf{I}\right)^{-1}\mathbf{H}_t^\top\boldsymbol{\Sigma}_t^{-1}R_{t-1} \\ =&\mathbf{U}_{t-1}\left(\mathbf{K}_{t-1} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{H}_t^{-1}R_{t-1} \\ =&\mathbf{U}_{t-1}\left(\mathbf{K}_{t-1}^x + \mathbf{U}_{t-1}^\top\mathbf{G}^{-1}\mathbf{U}_{t-1} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{H}_t^{-1}R_{t-1} \\ \approx&\mathbf{U}_{t-1}\left(\mathbf{K}_{t-1}^x + t\mathbf{P}_u + \sigma^2\mathbf{I}\right)^{-1}\mathbf{H}_t^{-1}R_{t-1} \end{aligned}$$

The 2nd line follows from Eq. 13, the 3rd is algebra, the 4th assumes that $t-1$ is the last step in an episode and therefore $\boldsymbol{\Sigma}_t^{-1} = \mathbf{H}_t^{\top\,-1}\mathbf{H}_t^{\,-1}/\sigma^2$, the 5th substitutes $\mathbf{K}_{t-1} = \mathbf{K}_{t-1}^x + \mathbf{U}_{t-1}^\top\mathbf{G}^{-1}\mathbf{U}_{t-1}$ where $\mathbf{K}_{t-1}^x$ is the kernel matrix corresponding to k_x and $\mathbf{U}_{t-1}^\top\mathbf{G}^{-1}\mathbf{U}_{t-1}$ is the kernel matrix corresponding to k_F. The last line is the result of replacing $\mathbf{G}$ with its estimator $\hat{\mathbf{G}}_{t-1}$. $\mathbf{P}_u = \mathbf{U}_{t-1}^\top(\mathbf{U}_{t-1}\mathbf{U}_{t-1}^\top)^{-1}\mathbf{U}_{t-1}$ is the projection operator on the span of $\{\mathbf{u}(\mathbf{z}_i)\}_{i=0}^{t-1}$.

As was pointed out in Engel et al. (2005), the i-th element of $\mathbf{H}_t^{-1}R_{t-1} = Y_t$ is $(Y_t)_i = \sum_{j=i}^{t-1}\gamma^{(j-i)}R(\mathbf{z}_j)$. Therefore, $\mathbf{E}\left[\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})|\mathcal{D}_t\right]$ is obtained by applying GP regression to samples of the discounted return, using the kernel matrix $\mathbf{K}_{t-1}^x + t\mathbf{P}_u$, and with $\mathbf{U}_{t-1}$ applied to the result. Taking the limit $\mathbf{K}_{t-1}^x \to \mathbf{0}$, leaves us with

$$\begin{aligned} \mathbf{E}\left[\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})|\mathcal{D}_t\right] &\approx \left(\sigma^2 + t\right)^{-1}\mathbf{U}_{t-1}\mathbf{H}_t^{-1}R_{t-1} \\ &= (\sigma^2 + t)^{-1}\sum_{i=0}^{t-1}\mathbf{u}(\mathbf{z}_i)(Y_t)_i, \end{aligned}$$

which is just a regularized (by the σ^2 term) version of the standard Monte-Carlo estimates for the $\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})$ as in Algorithms 1 and 2 of Baxter and Bartlett (2001).

Analogous analysis of the posterior covariance yields

$$\begin{aligned} &\mathbf{Cov}\left[\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})|\mathcal{D}_t\right] = \mathbf{G}_t - \mathbf{U}_t\mathbf{C}_t\mathbf{U}_t^\top \\ =&\mathbf{G}_t - \mathbf{U}_t\mathbf{H}_t^\top\left(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}\mathbf{H}_t\mathbf{U}_t^\top \\ =&\mathbf{G}_t - \mathbf{U}_t\left(\mathbf{H}_t^\top\boldsymbol{\Sigma}_t^{-1}\mathbf{H}_t\mathbf{K}_t + \mathbf{I}\right)^{-1}\mathbf{H}_t^\top\boldsymbol{\Sigma}_t^{-1}\mathbf{H}_t\mathbf{U}_t^\top \\ =&\mathbf{G}_{t-1} - \mathbf{U}_{t-1}\left(\mathbf{K}_{t-1} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{U}_{t-1}^\top \\ \approx&\hat{\mathbf{G}}_{t-1} - \mathbf{U}_{t-1}\left(\mathbf{K}_{t-1}^x + t\mathbf{P}_u + \sigma^2\mathbf{I}\right)^{-1}\mathbf{U}_{t-1}^\top \end{aligned}$$

Taking again the limit $\mathbf{K}_{t-1}^x \to \mathbf{0}$, results in

$$\begin{aligned} \mathbf{Cov}\left[\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})|\mathcal{D}_t\right] &\approx \hat{\mathbf{G}}_{t-1} - \left(t + \sigma^2\right)^{-1}\mathbf{U}_{t-1}\mathbf{U}_{t-1}^\top \\ &= \frac{\sigma^2}{\sigma^2 + t}\hat{\mathbf{G}}_{t-1}, \end{aligned}$$

which decays to 0 as $1/t$.

5. Experiments

In this section, we compare the Bayesian Actor-Critic (BAC), Bayesian Quadrature (BQ), and Monte-Carlo (MC) gradient estimates in a 10-state random walk problem. We also evaluate the performance of the BAC algorithm (Alg. 1) on the random walk problem, and compare it with a MC-based policy gradient (MCPG) algorithm (Baxter & Bartlett, 2001, Alg. 1), as well as a Bayesian policy gradient (BPG) algorithm (Ghavamzadeh & Engel, 2007, Alg. 2).

In the 10-state random walk problem, $\mathcal{X} = \{1, 2, \ldots, 10\}$, with states arranged linearly from state 1 on the left to state 10 on the right. The agent has two actions to choose from: $\mathcal{A} = \{right, left\}$. The left wall is a retaining barrier, meaning that if the $left$ action is taken at state 1, in the next time-step the state will be 1 again. State 10 is a zero reward absorbing state. The only stochasticity in the transitions is induced by the policy, which is defined as $\mu(right|\mathbf{x}) = 1/1 + \exp(-\theta_{\mathbf{x}})$ and $\mu(left|\mathbf{x}) = 1 - \mu(right|\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Note that each state $\mathbf{x}$ has an independent parameter $\theta_{\mathbf{x}}$. Each episode begins at state 1 and ends

when the agent reaches state 10. The mean reward is 1 for states 1–9 and is 0 for state 10. The observed rewards for states 1–9 are obtained by corrupting the mean rewards with a 0.1 standard deviation IID Gaussian noise. The discount factor is $\gamma = 0.99$.

We first compare the BAC, BQ1, BQ2, and MC estimates of $\nabla_{\boldsymbol{\theta}}\eta(\boldsymbol{\theta})$ for the policy induced by the parameters $\theta_{\mathbf{x}} = \log(4)$ for all $\mathbf{x} \in \mathcal{X}$, which is equivalent to $\mu(right|x) = 0.8$. The BQ1 and BQ2 gradient estimates were calculated using Model 1 and Model 2 from Ghavamzadeh and Engel (2007), respectively. We use several different sample sizes, measured by the number of episodes used in estimating the gradient: $M = 5j,\ j = 1, \ldots, 20$. For each value of M, we compute the gradient estimates 10^3 times, using the same data for all algorithms. The true gradient is calculated analytically for reference.

Fig. 1 shows the mean absolute angular error of the MC, BQ1, BQ2, and BAC estimates of the gradient for several different sample sizes M. The absolute angular error is the absolute value of the angle in degrees between the true gradient and the estimated gradient. In this experiment, the BAC gradient estimate was calculated using a Gaussian state kernel $k_x(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/(2\sigma_k^2))$, with $\sigma_k = 3$, and state-action kernel $0.01k_F(\mathbf{z}, \mathbf{z}')$. The results depicted in Fig. 1 indicate that the BAC gradient estimates are more accurate and have lower variance than their MC and BQ counterparts. We repeated this experiment by averaging over 10^3 different policies (instead of averaging over 10^3 runs of one policy) and the results were similar to those shown in Fig. 1, albeit with wider error bars. For lack of space we do not include these results here.

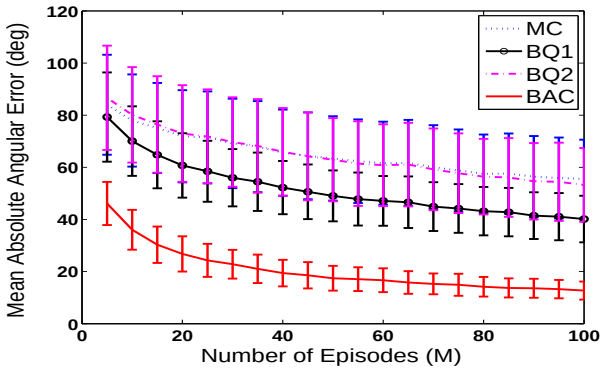

Figure 1. The mean absolute angular error of the MC, BQ1, BQ2, and BAC gradient estimations as a function of the number of sample episodes M. All results are averaged over 10^3 runs.

Next, we use BAC to optimize the policy parameters in the same random walk MDP. We compare the performance of BAC with a MCPG algorithm and a BPG algorithm, for $M = 1$, 5 and 20. The BPG algorithm uses Model 1 from Ghavamzadeh and Engel (2007). We use Alg. 1 with the number of policy updates set to 500 and the same kernels as in the previous experiment. The Fisher information matrix is estimated using $\hat{\mathbf{G}}$ from Eq. 23. The returns obtained by these methods are averaged over 100 runs.

For a fixed sample size M, each method starts with an initial learning rate and decreases it according to the schedule $\beta_t = \beta_0\beta_c/(\beta_c + t)$. We tried many values of the learning rate parameters (β_0, β_c) for MCPG, BPG, and BAC, and those in Table 1 yielded the best performance. $\beta_c = \infty$ means that we used a fixed learning rate β_0 for that experiment. Note that the learning rates used by the BAC algorithm are much larger than those used by the MCPG algorithm. This seems to be due to the fact that the BAC gradient estimates are more accurate than their MC counterparts. Since BPG is a path-based algorithm, the magnitude of the estimated gradients and therefore the learning rates used by BPG are in a different range than those estimated and used by the MCPG and BAC algorithms.

β_0 , β_c	$M = 1$	$M = 5$	$M = 20$
MCPG	0.25 , ∞	1.75 , 100	3 , ∞
BAC	5 , ∞	20 , 100	50 , 50
BPG	0.01 , 250	0.01 , 500	0.05 , ∞

Table 1. Values of the learning rate parameters used by the algorithms in the experiments of Fig. 2.

Fig. 2 depicts the results of these experiments. From left to right the sub-figures correspond to the experiment in which all algorithms used M=1, 5 and 20 trajectories per policy update, respectively. Each curve depicts the difference between the exact average discounted return for the 500 policies that follow each policy update and η^* – the optimal average discounted return. All curves are averaged over 100 repetitions of the experiment. The BAC algorithm clearly learns significantly faster than the other algorithms (note that the vertical scale is logarithmic).

6. Discussion

In this paper we presented a new Bayesian take on the familiar actor-critic architecture. By using Gaussian processes and choosing their prior distributions to make them compatible with a parametric family of policies, we were able to derive closed-form expressions for the posterior distribution of the policy gradient updates. The resulting algorithm addresses the

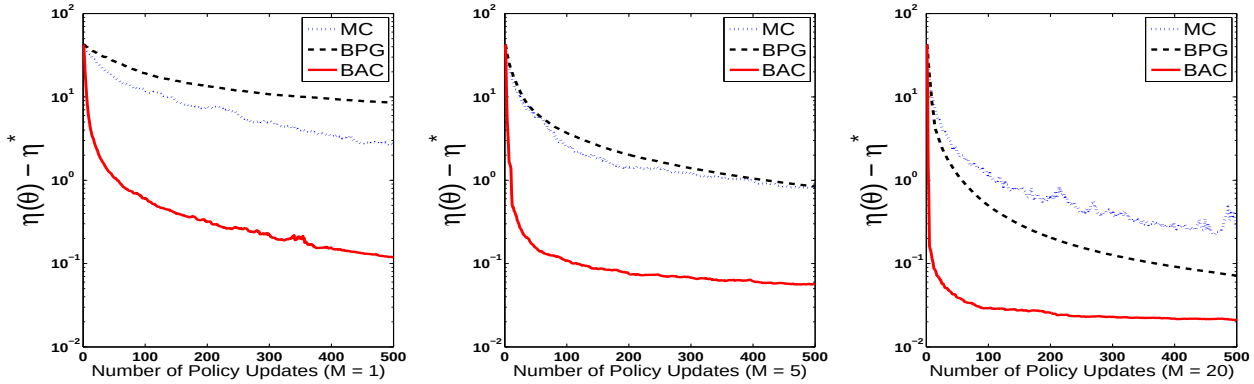

Figure 2. Results for the policy learning experiment. The graphs depict the performance of the policies learned by each algorithm during 500 policy update steps. From left to right the number of episodes used to estimate the gradient is $M = 1$, 5 and 20. All results are averaged over 100 independent runs.

main limitation of the Bayesian policy gradient algorithms of Ghavamzadeh and Engel (2007), which stems from them ignoring the Markov property of the system dynamics (when the system is indeed Markovian). This seems to be borne out in our experiments, where BAC provides more accurate estimates of the policy gradient than either of the two models proposed in Ghavamzadeh and Engel (2007), for the same amount of data.

As was done in the algorithms proposed in Engel et al. (2003); Engel et al. (2005); Ghavamzadeh and Engel (2007), the BAC algorithm can also be derived in a sparse form, which would typically make it significantly more time and memory efficient. For want of space we are not able to present a sparse variant of BAC here.

Additional experimental work is required to investigate the behavior of BAC in larger and more realistic domains, involving continuous and high-dimensional state spaces. The second-order statistics obtained from BAC are so far still unused. One interesting direction for future research would be aimed at finding a way to use the posterior covariance in determining the size and direction of the policy update.

Acknowledgments M.G. is supported by iCORE Canada. Y.E. is supported by an Alberta Ingenuity fellowship.

References

Baird, L. (1993). *Advantage updating* (Technical Report WL-TR-93-1146). Wright Laboratory.

Barto, A., Sutton, R., & Anderson, C. (1983). Neuron like elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man and Cybernetics, 13*.

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

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

Engel, Y. (2005). *Algorithms and representations for reinforcement learning*. Doctoral dissertation, The Hebrew University of Jerusalem, Israel.

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets Bellman: The Gaussian process approach to temporal difference learning. *Proceedings of the 20th International Conference on Machine Learning*.

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement learning with Gaussian processes. *Proceedings of the 22nd International Conference on Machine Learning*.

Ghavamzadeh, M., & Engel, Y. (2007). Bayesian policy gradient algorithms. *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press.

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

Konda, V., & Tsitsiklis, J. (2000). Actor-Critic algorithms. *Advances in Neural Information Processing Systems 12*.

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

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

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. *Advances in Neural Information Processing Systems 12*.