A Markov chain Monte Carlo algorithm for Bayesian policy search

Vahid Tavakol Aghaei^a, Ahmet Onat^a and Sinan Yıldırımb

^aFaculty of Engineering and Natural Sciences, Mechatronics Engineering, Sabancı University, Istanbul, Turkey; ^bFaculty of Engineering and Natural Sciences, Industrial Engineering, Sabancı University, Istanbul, Turkey

ABSTRACT
Policy search algorithms have facilitated application of Reinforcement Learning (RL) to dynamic systems, such as control of robots. Many policy search algorithms are based on the policy gradient, and thus may suffer from slow convergence or local optima complications. In this paper, we take a Bayesian approach to policy search under RL paradigm, for the problem of controlling a discrete time Markov decision process with continuous state and action spaces and with a multiplicative reward structure. For this purpose, we assume a prior over policy parameters and aim for the ‘posterior’ distribution where the ‘likelihood’ is the expected reward. We propound a Markov chain Monte Carlo algorithm as a method of generating samples for policy parameters from this posterior. The proposed algorithm is compared with certain well-known policy gradient-based RL methods and exhibits more appropriate performance in terms of time response and convergence rate, when applied to a nonlinear model of a Cart-Pole benchmark.

ARTICLE HISTORY
Received 6 March 2018
Accepted 22 September 2018

KEYWORDS
Reinforcement learning; Markov chain Monte Carlo; particle filtering; risk sensitive reward; policy search; control

1. Introduction

Reinforcement learning (RL) can be considered as a topic of interest among the researchers in the field of machine learning, control and robotics; see Sutton and Barto (1998) for an introduction. In RL, an agent interacts sequentially with an environment to collect some data samples and utilizes the generated data to find a policy that optimizes a long-term performance criterion. Among the existing RL methods, gradient-based algorithms have proved to be effective in dealing with high-dimensional state spaces. A drawback of these algorithms is that a local search of the policy is performed. Consequently, they may suffer from poor convergence rate or being trapped in local optima. In order to cope with this problem and facilitating convergence a Bayesian inference approach would be advocated.

This work concentrates on the control problem in Markov decision processes (MDP) with continuous state and action spaces in finite time horizon. We propose a novel RL policy search method using particle Markov chain Monte Carlo (P-MCMC), a recent and efficient family of MCMC methods for complex distributions, developed in Andrieu, Doucet, and Holenstein (2010). Our method is best applicable for risk-sensitive scenarios, where a multiplicative expected total reward is employed to measure the performance, rather than the more common additive one; since with a multiplicative reward one can fully utilize sequential Monte Carlo (SMC), aka the particle filter (Doucet, De Freitas, & Gordon, 2001; Gordon, Salmond, & Smith, 1993), within the iterations of the P-MCMC. The simulation results of learning to model and control a physical system reveal that significant improvements in time response of the plant as well as reduction in learning time can be achieved, which is important in real applications.

The proposed method does not require any gradient calculations and thus it does not generate a point estimate for the policy, instead, it works with estimates of the reward function directly and provides samples from the policy which can then be used to explore the surface of the policy performance to identify the favourable regions for the policy space. Therefore, it does not bear the risk of getting stuck in local optima or diverge due to poor choices of the learning rate for the gradient ascent update rules. In that sense, our method may be advantageous, at least in terms of breadth of applicability, over methods that do require gradient calculation. The claims on robustness and convergence time are supported with provided numerical experiments in Section 3.2 where the proposed method has been compared with two state-of-art gradient-based methods on the benchmark example of an inverted-pendulum problem. The comparisons are performed for two different reward structures, interval-based reward and quadratic...
reward. The simulation results regarding the convergence, which reveal the competitiveness of the suggested method with that of gradient-based ones are depicted in Figures 1 and 2.

The paper is organized as follows: In the rest of Section 1, we commence with a general description of MDPs, provide a short review of the gradient-based policy optimization strategies and proceed with Bayesian framework and risk-sensitivity ideas. In Section 2, the proposed method is presented. In order to demonstrate the efficiency of the proposed algorithm, numerical simulations are depicted in Section 3. Finally, concluding remarks and future work are expressed in Section 4.

1.1. Markov decision processes

RL problems can be specified with the notion of MDP. A Markov decision process is defined by the tuple 

\[(S, A, g, \eta, r)\] 

Here, \(S \subseteq \mathbb{R}^{d_s}, d_s > 0\), represents a set of \(d_s\)-dimensional continuous state space, and \(A \subseteq \mathbb{R}^{d_a}, d_a > 0\), stands for a continuous action space. We treat the state and action variables at time \(t\) as random variables \(S_t \in S\) and \(A_t \in A\) whose realizations will be denoted as \(s_t\) and \(a_t\). At time \(t > 0\), a transition from current state \(s_t\) to next state \(s_{t+1}\) as a result of taking an action \(a_t\) admits a transition law described by the transition density function \(g(s_{t+1}|s_t, a_t)\). We denote the probability density for the initial state by \(\eta(s_1)\). The reward function \(r: A \times S \times S \rightarrow \mathbb{R}\) assigns an immediate real-valued scalar reward also known as reinforce signal for the state transition from \(s_t\) to \(s_{t+1}\) with action \(a_t\), represented as \(r(a_t, s_t, s_{t+1})\).

Let the control policy be a stochastic parametrized policy with parameter \(\theta \in \Theta \subseteq \mathbb{R}^{d_\theta}\) for some \(d_\theta > 0\) and policy space \(\Theta\). The policy parameter \(\theta\) corresponds to a probability density function \(h_\theta(a_t|s_t)\) for the randomized action \(A_t\) at time \(t\) given state \(S_t = s_t\). Letting \(X_t = (S_t, A_t)\) taking values in \(\mathcal{X} = S \times A\), this induces a Markov chain for \(\{X_t\}_{t \geq 1}\) with transition law

\[f_\theta(x_{t+1}|x_t) := \mathcal{g}(s_{t+1}|s_t, a_t)h_\theta(a_t|s_t),\] (1)

where \(x_t = (s_t, a_t)\). Therefore, the joint probability density of a trajectory (also called path, or rollout) \(x_{1:n}\) until time

Figure 1. Performance comparison with respect to convergence – Interval-based reward. (a) MCMC, (b) eNAC, (c) GPOMDP, and (d) REINFORCE.
\[ p_{0}(x_{1:n}) := f_{0}(x_{1}) \prod_{t=1}^{n-1} f_{0}(x_{t+1}|x_{t}), \]  

where \( f(x_{1}) = \eta(s_{1})h_{0}(a_{1}|s_{1}) \), is the initial distribution for \( X_{1} \).

### 1.2. Reward-based policy search

The objective of policy search in RL is to seek optimal or plausible policy parameters \( \theta \) with respect to some expected performance of the trajectory \( X_{1:n} \). Define \( R_{n} : \mathcal{X}^{n} \rightarrow \mathbb{R} \) to be the discounted sum of the immediate rewards up to time \( n \)

\[ R_{n}(x_{1:n}) := \sum_{t=1}^{n-1} \gamma^{t-1} r(a_{t}, s_{t}, s_{t+1}), \]

where \( \gamma \in (0, 1] \) is a discount factor, and let \( U : \mathbb{R} \rightarrow \mathbb{R} \) be a monotonically increasing and continuous utility function with inverse \( U^{-1} \). In a finite horizon RL setting, the performance of a certain policy, \( J_{n}(\theta) \) based on \( U \) and \( R_{n} \) is given by

\[ J_{n}(\theta) = \mathbb{E}_{\theta}[U(R_{n}(X_{1:n}))] = \int p_{\theta}(x_{1:n}) U(R_{n}(x_{1:n})) \, dx_{1:n}, \]

where \( p_{\theta}(x_{1:n}) \) is the trajectory distribution. Various works formulate reinforcement learning as an inference problem for the policy parameter \( \theta \) that is based on either maximizing (some function of) \( J_{n}(\theta) \) with optimization techniques (Dayan & Hinton, 1997; Gullapalli, Franklin, & Benbrahim, 1994; Kappen, Gomez, & Opper, 2012; Kimura & Kobayashi, 1988; Maddison et al., 2017; Mitsunaga, Smith, Kanda, Ishiguro, & Hagita, 2005; Peters, 2005; Toussaint & Storkey, 2006), or exploring admissible regions of \( J_{n}(\theta) \) via some Bayesian approach (Hoffman, Doucet, Freitas, & Jasra, 2008; Wingate, Goodman, Roy, Kaelbling, & Tenenbaum, 2011).

### 1.3. Gradient-based algorithms for policy optimization

There are different choices for the function \( U \) depending on the nature of the reinforcement learning problem in terms of dealing with risk. A common choice is \( U(\chi) = \)
which corresponds to the risk-neutral case where the performance measure of a policy reduces to its expected total reward. This case has been most extensively studied in the literature. Specially, among the available RL methods, the policy gradient algorithms have drawn the most attention. The policy gradient algorithms can be implemented in high-dimensional state-action spaces. This makes them well-suited in the robotics domain, where indeed problems usually involve coping with aforementioned spaces. Usage of policy gradient algorithms has been pioneered in the works by Gullapalli et al. (1994). These methods have been employed to deal with complex control and robotics problems, such as those dealt in Kimura and Kobayashi (1988), Peters (2005), and Mitsunaga et al. (2005).

Specifically, the goal of policy optimization in RL is to quest optimal policy parameters \( \theta \) that maximizes the expected value of some function of \( R_n \).

\[
\hat{\theta} = \arg \max_{\theta \in \Theta} J_n(\theta).
\]  

(5)

Although it is hardly ever possible to evaluate \( \hat{\theta} \) directly with this choice of \( R_n \), maximization of \( J_n(\theta) \) can be accomplished by policy gradient (PG) methods that utilize the steepest ascent rule to update their parameters at iteration \( i \) as

\[
\theta^{(i+1)} = \theta^{(i)} + \beta \nabla J_n(\theta^{(i)}),
\]  

(6)

where \( \beta \) is a learning rate for the gradient ascent and the vector

\[
\nabla J_n(\theta) = \left( \frac{\partial J_n(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J_n(\theta)}{\partial \theta_d} \right)^T
\]

is the gradient of the expected total reward with respect to \( \theta = (\theta_1, \ldots, \theta_d) \). However, unless the state and the action spaces are finite or the Markov chain \( \{X_t\}_{t \geq 1} \) admit linear and Gaussian transitional laws, it is impossible or prohibitively difficult to evaluate the gradient \( \nabla J_n(\theta) \) in (6). In the sequel, we will review three main policy gradient methods that are proposed to efficiently approximate \( \nabla J_n(\theta) \). Those methods are the state-of-art methods and their performance will be compared with that of our algorithm in Section 3.

1.3.1. The REINFORCE algorithm

One of the very first methods in estimating \( \nabla J_n(\theta) \) in (6) is the REINFORCE algorithm introduced by Williams (1992) which exploits the idea of likelihood ratio methods. Since \( R_n \) does not depend on \( \theta \), \( \nabla J_n(\theta) \) can be written as:

\[
\nabla J_n(\theta) = \int \nabla p_\theta(x_{1:n}) R_n(x_{1:n}) \, dx_{1:n}.
\]  

(7)

Next, by using (2) as well as the ‘likelihood trick’ identified by \( \nabla p_\theta(x_{1:n}) = p_\theta(x_{1:n}) \nabla \log p_\theta(x_{1:n}) \), where the product

converted to summation according to logarithm’s specifications, we can rewrite

\[
\nabla J_n(\theta) = \int p_\theta(x_{1:n}) \left[ \sum_{t=1}^n \nabla \log h_\theta(a_t | s_t) \right] R_n(x_{1:n}) \, dx_{1:n}.
\]  

(8)

Due to the lack of exact information about the trajectory distribution \( p_\theta(x_{1:n}) \) or nonlinearity in \( p_\theta(x_{1:n}) \), the integration over this probability distribution may not be possible. The REINFORCE algorithm approximates \( \nabla J_n(\theta) \) by producing \( N \geq 1 \) generated trajectories of length \( n \) from \( p_\theta(x_{1:n}) \),

\[
x_{1:n}^{(i)} = (s_1^{(i)}, a_1^{(i)}, \ldots, s_n^{(i)}, a_n^{(i)}) \sim p_\theta(x_{1:n}), \quad i = 1, \ldots, N,
\]

and then performing the Monte Carlo estimate

\[
\nabla J_n(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left[ \sum_{t=1}^n \nabla \log h_\theta(a_t^{(i)} | s_t^{(i)}) \right] R_n(x_{1:n}^{(i)}). \]  

(9)

In Williams (1992), it is shown that this estimator suffers from high variance. In order to reduce this variance, Williams (1992) proposes to make use of a baseline \( b \in \mathbb{R}^{d_\theta} \) to modify (9) as

\[
\frac{\partial J_n(\theta)}{\partial \theta_j} \approx \frac{1}{N} \sum_{i=1}^N \left[ \sum_{t=1}^n \frac{\partial \log h_\theta(a_t^{(i)} | s_t^{(i)})}{\partial \theta_j} \right] \times \left( R_n(x_{1:n}^{(i)}) - b_j \right). \quad j = 1, \ldots, d_\theta.
\]  

(10)

This baseline \( b = (b_1, \ldots, b_{d_\theta}) \) is adaptively calculated during iterations from sample trajectories \( x_{1:n}^{(i)} \), in a heuristic manner so that the variance of the approximation is minimized and it is adapted during the iterations.

1.3.2. The G (PO) MDP algorithm

It can be observed from (9) that overall reward of a complete trajectory, \( R_n \), is used in order to calculate \( \nabla J_n(\theta) \). Due to the large variance in \( R_n(x_{1:n}) \), the gradient may not be estimated with high accuracy. In order to surmount this problem a modified version of the REINFORCE algorithm, called the G(PO)MDP algorithm, has been proposed by Baxter and Bartlett (2001), where the immediate reward of each time step is utilized as follows:

\[
\frac{\partial J_n(\theta)}{\partial \theta_j} \approx \frac{1}{N} \sum_{i=1}^N \left[ \sum_{t=1}^n \frac{\partial \log h_\theta(a_t^{(i)} | s_t^{(i)})}{\partial \theta_j} \right] \times \left( r(a_t^{(i)}, s_t^{(i)}, s_{t+1}^{(i)}) - b_t \right). \]  

(11)

for each \( j = 1, \ldots, d_\theta \), where the time varying baseline \( b_t = (b_{t,1}, \ldots, b_{t,d_\theta}) \) is also adaptively determined like the
1.3.3. The eNAC algorithm

Natural gradient methods introduced by Amari (1998) and Kakade (2002) have evolved into various policy gradient algorithms for RL as shown, such as Natural Actor Critic (NAC) and episodic NAC (eNAC) (Peters, 2005). The idea behind these algorithms is to explore $J_n(\theta)$ more efficiently by employing the Fisher information matrix for the trajectory distribution

$$F(\theta) = \mathbb{E}_\theta \left[ \nabla \log p(\theta)(X_{1:n}) \nabla \log p(X_{1:n})^T \right]$$  \hspace{1cm} (12)$$

and performing the gradient search with the natural gradient $F(\theta)^{-1} \nabla J_n(\theta)$ instead of $J_n(\theta)$. The eNAC algorithm estimates the natural gradient based on a regression that helps avoiding a direct approximation to $F(\theta)$. More details about the eNAC algorithm can be found in Peters (2005).

1.4. A Bayesian framework for RL

In this paper, we will deal with the particular choice that corresponds to risk-sensitive RL,

$$U(x) = \frac{1}{\kappa} \exp(\kappa x)$$  \hspace{1cm} (13)$$

for some risk factor $\kappa > 0$ (Howard & Matheson, 1972). For the particular choice of the utility function in (13), the performance measure $J_n(\theta)$ in (4) can be written explicitly as

$$J_n(\theta) = \mathbb{E}_\theta \left[ \frac{1}{\kappa} \exp(\kappa R_n(X_{1:n})) \right]$$

$$= \frac{1}{\kappa} \int p_\theta(x_{1:n}) \prod_{t=1}^{n-1} \exp \left\{ \kappa r(a_t, s_t, s_{t+1}) \gamma^{t-1} \right\} \, dx_{1:n}. $$  \hspace{1cm} (14)$$

The classical policy optimization problem in finite horizon risk sensitive setting can then be defined as finding the
policy $\theta \in \Theta$ that maximizes $U^{-1}(J_n(\theta))$, which is equivalent to maximizing $J_n(\theta)$, as noted in Marcus, Fernández-Gaucherand, Hernández-Hernández, Coraluppi, and Fard (1997).

Several risk-sensitive RL algorithms can be found in the literature. Risk sensitivity was originally discussed in early works in economics such as von Neumann and Morgenstern (1944). For RL, variants of the Q-learning approach (Watkins, 1989) or policy iteration (Sutton & Barto, 1998) can be found. Other examples for policy optimization for risk sensitive RL are Koenig and Reid Simmons (1994); Mihatsch and Neuneier (2002); Neuneier and Mihatsch (1998); Shen, Tobia, Sommer, and Obermayer (2014). However, these algorithms are for finite action spaces and therefore do not apply in general to our setting.

Lately, Maddison et al. (2017) consider the direct policy gradient approach for the choice in (13) that can be applied to general state and action spaces.

The main contribution in this paper is to propose a Bayesian approach for RL that is implemented via MCMC. Specifically, our methodology is a particle MCMC algorithm involving an SMC (particle filter) in its iterations. The novelty of the presented approach is due to a formulation of the policy search problem in a Bayesian inference where the expected multiplicative reward, is treated as a pseudo-likelihood function. The reason for taking $J_n(\theta)$ as an expectation of a multiplicative reward function is the ability to employ unbiased lower variance estimators of $J_n(\theta)$, despite the methods that utilize a cumulative rewards formulation which lead in estimates with high variance. We work with the choice in (13) for $U$ and hence Equation (14) is assumed for the performance measure of the policy parameter $\theta$. Instead of trying to come up with a single optimal policy, we cast the problem into a Bayesian framework where we treat $J_n(\theta)$ as if it is the likelihood function of $\theta$. Combined with an uninformative prior $\mu(\theta)$, this leads to a pseudo-posterior distribution for the policy parameter. We then aim to target a quasi-posterior distribution and draw samples from via applying MCMC which is constructed as

$$\pi_n(\theta) \propto \mu(\theta) J_n(\theta).$$

This approach may be desirable particularly because we can explore the high expected reward regions of $\theta$ without having to approximate any gradient information. Once we have samples from $\pi_n(\theta)$, we can deduct point estimates for $\theta$ if desired, such as the mean of the samples, or the maximum a posteriori point, etc.

Note that there are also Bayesian methods for policy learning tailored for the specific choice of $U(x) = x$. Similar to our setting, these methods assign a prior distribution with probability density $\mu(\theta)$ for the policy parameter $\theta$ and use some suitable function of $J_n(\theta)$ as the ‘likelihood’ function, therefore targeting the resulting posterior distribution for $\theta$, denoted as $\pi_n(\theta)$. For example, Hoffman et al. (2008) ensure a non-negative $J_n(\theta)$ and take that as the likelihood function, and then utilize a
trans-dimensional EM-based MCMC algorithm to sample from the posterior distribution that is proportional to

\[ \mu(\theta) J_n(\theta). \]  

(15)

The sampling procedure in their work is carried out by utilizing a reversible jump MCMC approach proposed in Green (1995), Green (2003) and Richardson and Green (1997). Drawing on the work of Hoffman et al. (2008), a hybrid MCMC algorithm was designed for (15) in Wilson, Fern, and Tadepalli (2010). Unlike Hoffman et al. (2008) and our method, the MCMC moves proposed in Wilson et al. (2010) targets an approximation of the posterior distribution by making use of Importance Sampling approaches (IS). Alternatively, Wingate et al. (2011) allows for negative reward but treats \( \exp(J_n(\theta)) \) as a ‘likelihood’ and then apply an approximate MCMC algorithm to sample from the posterior distribution that is proportional to \( \mu(\theta) \exp(J_n(\theta)) \).

There are also hybrid approaches such as the expectation–maximization (EM) algorithm in Toussaint and Storkey (2006), where they try to maximize the value function \( V(\theta) \) with respect to policy parameters \( \theta \) by applying forward-backward algorithms to apply E-step, where a fixed policy is assumed and posteriors are calculated over the state-action pairs. Afterwards the policy update procedure is done by M-step.

In the Bayesian methods for \( U(x) = x \) that are mentioned above, since \( J_n(\theta) \) is the expectation of an additive function of instantaneous rewards, estimating \( J_n(\theta) \)

---

**Figure 5.** A sample time response for stabilization of the Cart-Pole – Interval-based reward. (a) Angle of the pendulum, (b) position of the cart, and (c) control signal.
and incorporating the estimate in the methods seem to be challenging due to high variance, difficulty in ensuring non-negative $J_n(\theta)$, or bias as shown by Maddison et al. (2017). In the next section, we will show that in the risk-sensitive scenario, where $J_n(\theta)$ is the expectation of a multiplicative quantity (see (14)), we can utilize efficient, lower variance, unbiased estimators of $J_n(\theta)$ in our proposed method.

2. MCMC for policy search

MCMC methods are a vastly popular family of methods used to obtain samples from complex distributions. Here, the distribution of our interest is $\pi_n(\theta)$, which is indeed hard to calculate expectations with respect to or generate exact samples from.

MCMC methods are based on generating an ergodic Markov chain $\{\theta(k)\}_{k \geq 0}$, starting from the initial point $\theta(0)$, which has the desired distribution, in our case $\pi_n(\theta)$, as its invariant distribution. Arguably the most widely used MCMC method is the Metropolis–Hastings (MH) method, which works as follows: For $k = 1, 2, \ldots$, given $\theta^{(k-1)} = \theta$, we sample a proposal value $\theta'$ from a proposal distribution $q(\cdot|\theta)$ conditional on $\theta$, i.e.

$$\theta' \sim q(\theta'|\theta).$$

Then, we either accept the proposal and set $\theta^{(k)} = \theta'$ with probability $\alpha(\theta, \theta') = \min\{1, \rho(\theta, \theta')\}$ or we reject the proposal and set $\theta^{(k)} = \theta$, where the acceptance ratio is

$$\rho(\theta, \theta') = \frac{q(\theta'|\theta) \pi_n(\theta')}{q(\theta'|\theta) \pi_n(\theta)} = \frac{q(\theta'|\theta) \mu(\theta') J_n(\theta')}{q(\theta'|\theta) \mu(\theta) J_n(\theta')}.$$

The MH algorithm requires the ability of calculating the acceptance ratio $\rho(\theta, \theta')$ for any given $\theta, \theta' \in \Theta$. For our case, this is not possible since calculation of $\rho(\theta, \theta')$ involves calculation of $J_n(\theta)$ and $J_n(\theta')$ which is not generally possible due to the complex structure of $J_n(\theta)$ in (14). Fortunately, we can still target $\pi_n(\theta)$ using an MCMC algorithm if unbiased and non-negative estimates

![Figure 6](image-url)
of \( J_n(\theta) \) can be obtained, i.e. we can obtain random variables \( \hat{J}_n(\theta) \geq 0 \) such that

\[
\mathbb{E}[\hat{J}_n(\theta)] = J_n(\theta).
\]

The MH algorithm that uses such \( \hat{J}_n(\theta) \) instead of \( J_n(\theta) \) is called the pseudo-marginal MH algorithm (Andrieu & Roberts, 2009); see also Andrieu et al. (2010) for a broader family of such algorithms for hidden Markov models, namely particle MCMC algorithms. The implementation of the pseudo-marginal MH algorithm for our RL problem is given in Algorithm 1.

**Algorithm 1:** Pseudo-marginal MH for RL

**Input:** Number of time steps \( n \), initial value and estimate of expected performance \((\theta(0), \hat{J}_n(0))\), proposal distribution \( q(\theta'|\theta) \), number of particles \( N \)

**Output:** Samples \( \theta^{(k)} \), \( k = 1, 2, \ldots \)

for \( k = 1, 2, \ldots \) do

- Given \( \theta^{(k-1)} = \theta \) and \( \hat{J}_n^{(k-1)} = \hat{J}_n \), sample a proposal value \( \theta' \sim q(\theta'|\theta) \).
- Obtain an unbiased estimate \( \hat{J}_n \) of \( J_n(\theta') \) by Algorithm 2 with \( N \) particles.
- Accept the proposal and set \( \theta^{(k)} = \theta' \) and \( \hat{J}_n^{(k)} = \hat{J}_n \) with probability \( \min\{1, \hat{\rho}(\theta, \theta')\} \)
  \[
  \hat{\rho}(\theta, \theta') = \frac{q(\theta'|\theta)}{q(\theta'|\theta')} \frac{\mu(\theta')}{\mu(\theta)} \frac{\hat{J}_n}{J_n},
  \]
  otherwise reject the proposal and set \( \theta^{(k)} = \theta \) and \( \hat{J}_n^{(k)} = \hat{J}_n \).

end

### 2.1. SMC for estimating \( J_n(\theta) \)

Algorithm 1 displays we can work with non-negative unbiased estimators of \( J_n(\theta) \) itself instead of \( J_n(\theta) \) itself, and still target \( \pi_n(\theta) \) exactly. The question is, then, how to obtain unbiased estimators of \( J_n(\theta) \) for any given \( \theta \). This is possible using an SMC algorithm thanks to the fact that the reward function is multiplicative. For sake of simplicity (and without loss of generality regarding the validity of our methodology), we will assume from now on that \( \kappa = 1 \) and the instantaneous reward function only depends on \( x_t = (a_t, s_t) \) and not \( s_{t+1} \). With this assumption, we can rewrite

\[
J_n(\theta) = \int f_\theta(x_1) \exp\{r(a_1, s_1)\} \prod_{t=1}^{n-1} f_\theta(x_t|x_{t-1}) \exp\{\gamma^{t-1}r(a_t, s_t)\} \, dx_1 x_n,
\]

which suggests that we may cast the problem of estimating the expected overall reward in our RL setting into the framework of Feynman–Kac formulae (DelMoral, 2004), in which \( \{X_t\}_{t \geq 1} \) is the Markov chain and \( \exp\{\gamma^{t-1}r(a_t, s_t)\} \) is the potential function at time \( t \). In other words, we can think of \( \{X_t\}_{t \geq 1} \) as the latent Markov process of a hidden Markov model, and \( \gamma^{t-1}r(a_t, s_t) \) as the conditional observation density at time \( t \) for some fixed observation. It was shown in DelMoral (2004) that a non-negative and unbiased estimator of \( J_n(\theta) \) can be obtained by running an SMC algorithm, aka a particle filter, as in Algorithm 2.

**Algorithm 2:** SMC algorithm for an unbiased estimate of \( J_n(\theta) \)

**Input:** Policy \( \theta \), number of time steps \( n \), discount factor \( \gamma \)

**Output:** Unbiased estimate of \( \hat{J}_n \)

Start with \( \hat{J}_0 = 1 \).

for \( t = 1, \ldots, n \) do

- for \( i = 1, \ldots, N \) do

  - if \( t = 1 \) then
    - Sample \( \hat{X}_t^{(i)} \sim \eta_\theta(x_1) \).
    - else
    - Sample \( \hat{X}_t^{(i)} \sim f_\theta(\cdot|X_{t-1}^{(i)}) \) using (1).
  - end

- Calculate \( W_t^{(i)} = e^{t-1}r(\hat{X}_t^{(i)}) \).

end

if \( t < n \) then

- Resample from \( \{\hat{X}_t^{(i)}, i = 1, \ldots, N\} \) with probabilities proportional to \( \{W_t^{(i)}, i = 1, \ldots, N\} \) to obtain resampled particles \( \{X_t^{(i)}, i = 1, \ldots, N\} \), i.e.

\[
P(X_t^{(i)} = \hat{X}_t^{(i)}) = \frac{W_t^{(i)}}{\sum_{j=1}^{N} W_j^{(i)}}, \quad i = 1, \ldots, N.
\]

end

Update the estimate: \( \hat{J}_t = \hat{J}_{t-1} \times \frac{1}{N} \sum_{i=1}^{N} W_t^{(i)} \)

return \( \hat{J}_n \).

To put our methodology in context, it is worth acknowledging the works that relate reinforcement learning and hidden Markov models (or more generally, Feynman–Kac formulae) and thereby adopting existing inference tools developed for hidden Markov models to reinforcement learning already: Toussaint and Storkey (2006) developed an EM algorithm for finite and Gaussian reinforcement learning scenarios. Rawlik, Toussaint, and Vijayakumar (2010) proposed an
approximate inference approach, again based on the EM algorithm from more challenging scenarios. Finally, a recent independent work by Maddison et al. (2017) points to this relation and proposes Algorithm 2 for an unbiased estimator of $J_n(\theta)$, as we do. However, while Maddison et al. (2017) ultimately uses this estimator to utilize a policy gradient algorithm for risk-sensitive policy learning, we employ it in an MCMC algorithm by exploring the posterior distribution for Bayesian learning. To the best of our knowledge, no previous work has targeted $\pi(\theta)$ by executing an SMC algorithm in a Bayesian framework for a multiplicative reward structure, without the need to calculate gradient information.

Algorithm 2 requires propagation, weighting, and resampling of $N$ particles at a time step. When only one particle is used i.e. $N=1$, the algorithm reduces to sampling a single path $X_{1:n} \sim p_\theta(X_{1:n})$ and calculating $\exp\{R(X_{1:n})\} = \prod_{t=1}^{n} e^{\gamma t - 1 r(X_t)}$ as an estimate of the expected reward. Despite its expected inferior statistical properties compared to $N>1$, the $N=1$ version of Algorithm 2 is worth noting, because it allows for a straightforward real time implementation of the algorithm, say, by a single robot.

3. Simulations

This section discusses the implementation of the proposed method for a nonlinear model of an inverted pendulum and demonstrates its usefulness in comparison with policy gradient algorithms for reinforcement learning.

3.1. The inverted pendulum model

The state space $S$ of the inverted pendulum is constituted of four continuous components that are angle, angular velocity, position and velocity of the cart-pole system, i.e. $s_t = (\phi_t, \dot{\phi}_t, p_t, \dot{p}_t)$. The action space $A$ is one-dimensional and continuous which plays the role of the force applied to the cart. It is assumed that the previous state and the current action determine the current state, $s_t = G(s_{t-1}, a_t)$, where $G: S \times A \rightarrow S$ is a deterministic function, so that $g(s_t|s_{t-1}, a_t) = \delta_{G(s_{t-1}, a_t)}(s_t)$ in terms of densities. The function $G$ is implicitly dictated by the dynamics of the inverted pendulum, given by Dann, Neumann, and Peters (2014). When action $a$ is applied to the pendulum at state $s = (\phi, \dot{\phi}, p, \dot{p})$, the acceleration in $\phi$ and $p$ are determined as

$$
\ddot{\phi} = \frac{-3m\ell^2 \sin \phi \cos \phi + 6(M+m)g \sin \phi}{-6(a - b\dot{\phi}) \cos \phi - 4l(M+m) - 3ml \cos \phi}
$$

(16a)
\[ \ddot{\phi} = \frac{-2mL^2 \sin \phi + 3mg \sin \phi \cos \phi + 4a - 4b \dot{\phi}}{4(M + m) - 3m \cos \phi} \] (16b)

where \( g = 9.8 \, \text{m/s}^2 \) is the gravitational constant, \( b = 0.1 \, \text{N/ms} \) is the friction coefficient, \( L = 0.6 \, \text{m} \) is the length of the pole, \( M = 0.5 \, \text{kg} \) is the mass of the cart, and \( m = 0.5 \, \text{kg} \) is the mass of the pole. The control objective is to stabilize the inverted pendulum in the upright position as well as to bring the cart to the origin from a random initial state.

Given the state value \( S_t = s_t \) at time \( t \), the action \( A_t \) is stochastic and drawn from the Gaussian distribution \( \mathcal{N}(\theta^T s_t, \sigma^2) \) with mean \( \theta^T s_t \) and variance \( \sigma^2 \) so that the conditional density of the action is,

\[ h_{\theta}(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(a - \theta^T s)^2}{2\sigma^2} \right\}. \] (17)

The objective of policy search is to learn the parameter vector \( \theta \), while the variance in action is assumed to be constant.

### 3.2. Numerical experiments

The performance of the proffered method will be compared with three algorithms of policy gradient RL, namely eNAC, REINFORCE and GPOMDP. During the simulations two distinct structured reward functions for the proposed method and PG algorithms have been utilized. The
Reward functions here are defined as a delayed reward where in our numerical simulations we call it Interval-based reward

\[
r(a, s) = \begin{cases} 
0, & -12^\circ \leq \phi \leq 12^\circ, \text{ and} \\
-10, & \text{otherwise.}
\end{cases}
\]

(18)

as well as, a quadratic reward function

\[
r(a, s) = -s^T Q s - c a^2,
\]

(19)

where \( Q \) is a positive definite matrix and \( c \) is a positive real number. Here, we take \( Q \) to be a diagonal matrix with the diagonal \([12, 1.25, 2, 0.25]^T\) and \( c = 0.01 \). These predefined values are beneficial in adjusting the contributions of the state and action to the reward function. Discounted factor for the overall reward function \( R_n \) is \( \gamma = 0.99 \). The simulation length for all of the experiments is \( n = 1000 \) time steps with a sampling time of 0.01 in seconds. During the learning process, the variance of the action is taken as \( \sigma^2 = 4 \); however, during the testing process, this value is set to 0 to reflect a deterministic action policy.

| Reward type | MCMC  | eNAC  | GPOMDP | REINFORCE |
|-------------|-------|-------|--------|-----------|
| Interval based | 1256.1368 | 821.7421 | 2210.0986 | 2349.0957 |
| Quadratic | 378.1922 | 418.8248 | 454.8195 | NA \(^a\) |

\(^a\) For Quadratic reward, REINFORCE did not converge to an optimal solution.

Figure 9. Parameter update for MCMC with quadratic rewards regarding different number of particles. (a) MCMC with \( N_p = 10 \) and (b) MCMC with \( N_p = 100 \).

Figure 10. Performance for MCMC with quadratic rewards regarding different number of particles. (a) MCMC with \( N_p = 10 \) and (b) MCMC with \( N_p = 100 \).
3.2.1. Comparison with gradient-based methods based on total discounted reward

First, we compare our MCMC method with three gradient-based methods, namely REINFORCE, GPOMDP, and eNAC as mentioned in Section 1.3. Each of the gradient-based methods is run for 500 iterations and $H=100$ episodes are run at each iteration to calculate an approximation of the gradient. For the MCMC method in Algorithm 1, 50,000 iterations are run, with $N=1$ particle for each iteration. This choice of algorithm parameters for the compared methods ensures that the computational complexity for all the methods is the same. Also, a Gaussian random walk proposal is chosen with $q(\theta'|\theta) = \mathcal{N}(\theta'; \theta, \Sigma_q)$, where $\Sigma_q$ is a diagonal covariance matrix with $(10, 1, 1, 1)^T$ on its diagonal. The prior distribution for $\theta$ is $\mathcal{N}(0, 10^2I_4)$, which reflects the lack of prior information for $\theta$ prior to our runs. Here $I_4$ is a $4 \times 4$ identity matrix. Later we will discuss the importance of the choice of $\Sigma_q$ for a better performance of the algorithm by setting two different values for $\Sigma_q$ and inspect the resulting effects.

3.2.1.1. Interval-based reward. Firstly, we performed comparisons according to the interval-based reward function represented in Equation (18). The corresponding results depicted in Figure 3, where the trace plots for parameter updates versus simulation time for the suggested MCMC algorithm and gradient-based methods are given. While the gradient-based methods aim for a point estimate, the MCMC method aims for the quasi-posterior $\pi_n(\theta)$ in a Bayesian framework and provides samples for $\theta$ that are (approximately) distributed from $\pi_n(\theta)$. We can observe from plots for the gradient methods that those methods, converge to different maximal points of the expected reward function. However, samples from the MCMC methods ‘recognize’ both of these maximal points by exploring (roughly) the high-probability regions of $\pi_n(\theta)$. This can be better seen from the histograms in Figure 4 for the components of $\theta$, representing the component-wise marginal distributions with respect to $\pi_n(\theta)$.

Next, we measure the differences between the performances of the proposed method and PG algorithms in terms of convergence. For this purpose, the simulations are run 5 times each and averaged over estimated expected total rewards, where for the MCMC method, the logarithms of the estimated expected overall rewards, i.e. $\log \hat{J}_n(\theta)$ versus simulation time are observed. This is demonstrated in Figure 1 where the shaded parts in the plots show the standard deviation. As can be observed from Figure 1, expected average return for MCMC has converged in a faster manner in comparison to others. Note that $J_n(\theta)$ is taken differently for the MCMC (with utility function $U(x) = \exp(x)$) and the gradient-based methods (with utility function $U(x) = x$), which explains the difference in the values at the y-axes; however, the convergence time argument is still justified.

To have a measure of the performance of the time responses of the obtained parameters, we calculated the composition of the Integral Absolute Error (IAE) for $\phi_t$, $p_t$, and $a_t$ (the closer each of them to 0, the better) as

$$\text{IAE} = \int_0^\infty |e_t|dt$$

where $e_t$ stands for the error signal coming from the difference between the desired values of $\phi_t$, $p_t$, and $a_t$ with their actual quantities in the feedback control. While the $\theta$ values selected to measure the IAE for the gradient-based methods are simply the values obtained at the last iteration, for MCMC method, this is the average of last quarter of the total 50,000 iterations of the algorithm, representing an estimate for the mean of the target distribution $\pi_n(\theta)$. Figure 5 illustrates both obtained IAE and a sample time response for the compared methods revealing that performance of the proposed method outperforms GPOMDP and REINFORCE based on IAE in the time domain responses; however it is outperformed by eNAC, when an Interval-based reward function is used.

3.2.1.2. Quadratic reward. Here we replaced the interval-based reward function with a quadratic one provided in (19) and repeated the experiments. It is worth mentioning that quadratic rewards are widely used in learning the behaviour of the dynamic systems. As before, the trace plots for MCMC and gradient-based methods are generated and then presented in Figure 6, histograms for the samples obtained from MCMC are given in Figure 7, estimated expected returns are depicted in Figure 2 and finally both the IAE, as well as, a sample time response for the compared methods are made available in Figure 8. Note that for the quadratic reward case REINFORCE algorithm could not manage to converge to an optimal solution, that is why it is excluded from the comparisons. The conclusions that can be drawn from those figures for this reward type are almost the same as for interval-based reward, except that as it is obvious from Figure 8, MCMC performed better than eNAC and GPOMDP in terms of the IAE for the time responses, as well. The control performance in terms of IAE is summarized in Table 1. Note that $x$-axes in part (a) for both Figures 1 and 2 are in the same range as other parts of these figures but for a clear representation of the early convergence of the performance figures of MCMC, we zoomed in a given portion of the $x$-axes.
It is worth noting that the structure of the reward function influences the ultimate performance of the control system in terms of time response. The situation can be better understood by comparing Figures 5 and 8 where using a Quadratic reward improves the overall time responses of the system by reducing the settling time as well as eliminating the overshoots occurred (thereby resulting in a less IAE) in time responses which used an Interval-based reward.

3.2.2. Comparison within the MCMC method

3.2.2.1. Effects of having different number of particles.

By taking into account the real time applications of the proposed method, for all the above experiments, MCMC has been performed by taking number of particles as $N_p = 1$. In simulation environments we can use different number of particles in order to improve the performance of the algorithm.

![Figure 11](image-url)

**Figure 11.** Histogram of the estimated parameters for MCMC with quadratic rewards regarding different number of particles. (a) MCMC with $N_p = 10$ and (b) MCMC with $N_p = 100$. 
By utilizing the quadratic reward structure, the performance of the proposed MCMC method for $N_p = 10$ and $N_p = 100$ particles, are compared with each other. Sampled values for the policy parameters, performance in terms of the total expected reward and histograms for the learned policy parameters are demonstrated in Figures 9–11, respectively. As one can observe from the figures, $N_p = 100$ produces smoother histogram estimates for the marginal distributions. However, convergence of the algorithm needs more time in case of $N_p = 100$ due to a 10-fold increase in computation time per iteration, as it is obvious from Figure 10.

3.2.2.2. Effects of the proposal covariance $\Sigma_q$ and action variance $\sigma^2$. Samples generated by the MCMC need to be mixed well in a way that they easily forget their previous values. In particular, we need to choose a suitable proposal variance $\Sigma_q$ to ensure well-mixing. While a very small $\Sigma_q$ will result in providing samples that are highly correlated and mixing in a very slow fashion, a very large $\Sigma_q$ is prone to stick to the old values of the generated samples by rejecting the new proposals. Below, we demonstrate this phenomenon in the policy search setting. We take a quadratic reward structure as in the previous subsection and we looked at all the four combinations of $(\sigma, \Sigma_q)$ for

- $\sigma = 2.5, \sigma = 1.5$ and
- $\Sigma_q = \text{diag}[100, 10, 10, 10], \Sigma_q = \text{diag}[1, 0.1, 0.1, 0.1]$

The corresponding simulations to these settings are depicted in Figures 12 and 13. When comparing the trace

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure12.png}
\caption{Policy parameters and performances of MCMC considering small and large values for $\Sigma_q$ based on the quadratic reward when $\sigma = 2.5$. (a) Trace plots generated with a large $\Sigma_q$. (b) Trace plots generated with a small $\Sigma_q$. (c) Performance of the MCMC with a large $\Sigma_q$. (d) Performance of the MCMC with a small $\Sigma_q$.}
\end{figure}
Figure 13. Policy parameters and performances of MCMC considering small and large values for $\Sigma_q$ based on the quadratic reward when $\sigma = 1.5$. (a) Trace plots generated with a large $\Sigma_q$. (b) Trace plots generated with a small $\Sigma_q$. (c) Performance of the MCMC with a large $\Sigma_q$. (d) Performance of the MCMC with a small $\Sigma_q$.

plots in those figures with the one in Figure 6(c), it is observed that choosing $\Sigma_q$ to be neither very large nor very small, as in Figure 6(c), results in the method to converge to its target distribution in a way that all the generated samples are mixed well. Specifically, part (a) of Figures 12 and 13 indicate that the large values for $\Sigma_q$ may cause the rejection of the proposed quantities for the policy parameters and the generated samples will have a tendency to stick to their old values. In part (b) of the same figures, the generated samples mix slowly and therefore are highly correlated due to small $\Sigma_q$.

We also observe that, in spite of bad choices of $\Sigma_q$, the proposed MCMC algorithm is still able to tackle the policy parameter estimation problem and manages to converge, where their performance in case of convergence properties are presented in parts (c) and (d) of Figures 12 and 13.

Finally, our conclusions hold for the two different values of the action variance $\sigma$, which indicates a certain level of robustness for our method.

4. Conclusion and future work

In this paper, we considered the control problem of an MDP with continuous state and action spaces in finite time horizon and presented a novel policy search method for a class of parameterized polices in RL under a Bayesian learning framework. The proposed learning approach is an MCMC algorithm that uses estimates of the expected multiplicative total reward which are attained with an
SMC algorithm. Using a multiplicative reward formulation, inspired by risk-sensitivity notion, proved to be advantageous in comparison to its additive counterpart as illustrated by our simulation results.

The proposed MCMC algorithm was applied to the control problem of stabilization of a nonlinear model of an inverted pendulum using both quadratic and delayed reward, and simulation results were compared with some well known gradient-based policy search (PG) approaches. Comparing the results achieved by the proposed MCMC algorithm with those of PG algorithms, some observations stand out:

(i) Whereas the PG methods attempt to find a point estimate for the policy parameters (and are prone to get trapped in local optima), the proposed MCMC algorithm aims for the quasi-posterior distribution in a Bayesian scheme which yields admissible values for the policy parameter. In the advanced Bayesian policy search approach, since the objective distribution is proportional to the expected total reward, then search for the optimal parameters are steered to the regions with high probable reward portions.

(ii) The proposed method either outperforms the tested PG algorithms in terms of time response specifications, or its performance is close enough to that of PG methods and meets the desired control objectives.

(iii) When it comes to comparing the performance in terms of convergence time, the proposed MCMC algorithm was observed to be superior to the tested PG algorithms.

(iv) Proposed MCMC algorithm can be advantageous over the gradient-based methods by circumventing the difficulties faced when using gradient information. Gradient-based tactics may be afflicted by poor convergence performances by getting stuck in local optima or diverging due to the weak choices of the learning rate used in gradient ascent updating rules, instead we suggest to leverage estimates of the expected multiplicative reward, without the need to approximate any gradient.

In this work, we show the merits of the proposed MCMC algorithm in Algorithm 1 via computer simulations. In a real-time system, it may be hard to implement, for example with a robot, when multiple particles are involved at the same time. However, for such cases, Algorithm 1 can still be used effectively with one particle per iteration, as shown in the numerical section. Indeed, the work is underway to use the $N = 1$ scheme for a real-world discrete time implementation where trajectories are generated by a robot.

Note

1. For the general case of dependency on $s_{t+1}$ as well, one can extend the state $X_t$ of the Markov chain to include $s_{t+1}$ and redefine the transition laws of the modified Markov chain, then the methodology developed in this paper can also be applied to the modified Markov chain.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2), 251–276.

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3), 269–342.

Andrieu, C., & Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2), 697–725.

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

Dann, C., Neumann, G., & Peters, J. (2014). Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15(1), 809–883.

Dayan, P., & Hinton, G. (1997). Using expectation–maximization for reinforcement learning. *Neural Computation*, 9, 271–278.

Del Moral, P. (2004). *Feynman-Kac formulae: Genealogical and interacting particle systems with applications*. New York, NY: Springer-Verlag.

Doucet, A., De Freitas, J., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice*. New York, NY: Springer-Verlag.

Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings F*, 140(6), 107–113.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4), 711–732.

Green, P. J. (2003). Trans-dimensional Markov chain Monte Carlo. In P. J. Green, N. L. Hjort, & S. Richardson (Eds.), *Highly structured stochastic systems* (pp. 179–198). Oxford: Oxford University Press.

Gullapalli, V., Franklin, J. A., & Benbrahim, H. (1994). Acquiring robot skills via reinforcement learning. *IEEE Control Systems*, 14(1), 13–24.

Hoffman, M., Doucet, A., Freitas, N. D., & Jasra, A. (2008). Bayesian policy learning with trans-dimensional MCMC. In J. C. Platt, D. Koller, Y. Singer, & S. T. Roweis (Eds.), *Advances in neural information processing systems 20* (pp. 665–672). Red Hook, NY: Curran Associates, Inc.

Howard, R. A., & Matheson, J. E. (1972). Risk-sensitive Markov decision processes. *Management Science*, 18(7), 356–369.

Kakade, S. (2002). A natural policy gradient. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing Systems 14* (pp. 1531–1538). Cambridge, MA: MIT Press.

Kappen, H. J., Gomez, V., & Opper, M. (2012). Optimal control as a graphical model inference problem. *Machine Learning*, 87, 159–182.
Kimura, H., & Kobayashi, S. (1988). Reinforcement learning for continuous action using stochastic gradient ascent. The 5th International Conference on Intelligent Autonomous Systems, (pp. 288–295).

Koenig, S., & Simmons, R. G. (1994). Risk-sensitive planning with probabilistic decision graphs. In Proceedings of the 4th international conference on principles of knowledge representation and reasoning (pp. 363–373).

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Doucet, A., Mnih, A., & Teh, Y. W. (2017). Particle value functions. Workshop track – ICLR 2017.

Marcus, S. I., Fernández-Gaucherand, E., Hernández-Hernández, D., Coraluppi, S., & Fard, P. (1997). Risk sensitive Markov decision processes (pp. 263–279). Boston, MA: Birkhäuser Boston.

Mihatsch, O., & Neuneier, R. (2002). Risk-sensitive reinforcement learning. Machine Learning, 49(2), 267–290.

Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., & Hagita, N. (2005). Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. In IEEE/RSJ International conference on intelligent robots and systems (IROS), Edmonton, Alta (pp. 1594–1601). IEEE.

Neuneier, R., & Mihatsch, O. (1998). Risk sensitive reinforcement learning. In Proceedings of the 11th international conference on neural information processing systems, NIPS’98 (pp. 1031–1037). Cambridge, MA: MIT Press.

Peters, J. (2005). Machine learning of motor skills for robotics (Phd thesis Proposal USC Technical Report, Tech Rep).

Rawlik, K., Toussaint, M., & Vijayakumar, S. (2010). An approximate inference approach to temporal optimization in optimal control. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta, (Eds.), Advances in neural information processing systems 23 (pp. 2011–2019). Red Hook, NY: Curran Associates, Inc.

Richardson, S., & Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components. Institute of International Economics Project on International Competition Policy, QUOT; COM/DAFFE/CLP/TD(94)42.

Shen, Y., Tobia, M. J., Sommer, T., & Obermayer, K. (2014). Risk-sensitive reinforcement learning. Neural Computation, 26(7), 1298–1328.

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction [Adaptive computation and machine learning]. Cambridge, MA: MIT Press.

Toussaint, M., & Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state markov decision processes. In Proceedings of the 23rd international conference on machine learning, ICML ’06, (pp 945–952). New York, NY: ACM.

von Neumann, J., & Morgenstern, O. (1944). Theory of games and economic behavior. Princeton: Princeton University Press.

Watkins, C. J. C. H. (1989). Learning from delayed rewards (PhD thesis, King’s College). Cambridge, UK.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3–4), 229–256.

Wilson, A., Fern, A., & Tadepalli, P. (2010). Bayesian role discovery for multi-agent reinforcement learning. In 9th International conference on autonomous agents and multiagent systems (AAMAS 2010), Toronto, Canada, May 10–14, 2010, Volume 1–3 (pp. 1587–1588).

Wingate, D., Goodman, N. D., Roy, D. M., Kaelbling, L. P., & Tenenbaum, J. B. (2011). Bayesian policy search with policy priors. In Proceedings of the twenty-second international joint conference on artificial intelligence (pp. 16–22).