Linear Bayesian Reinforcement Learning

Nikolaos Tziortziotis
ntziorzi@gmail.com
University of Ioannina

Christos Dimitrakakis
christos.dimitrakakis@gmail.com
EPFL

Konstantinos Blekas
kblekas@cs.uoi.gr
University of Ioannina

Abstract

This paper proposes a simple linear Bayesian approach to reinforcement learning. We show that with an appropriate basis, a Bayesian linear Gaussian model is sufficient for accurately estimating the system dynamics, and in particular when we allow for correlated noise. Policies are estimated by first sampling a transition model from the current posterior, and then performing approximate dynamic programming on the sampled model. This form of approximate Thompson sampling results in good exploration in unknown environments. The approach can also be seen as a Bayesian generalisation of least-squares policy iteration, where the empirical transition matrix is replaced with a sample from the posterior.

1 Introduction

Reinforcement learning is the problem of learning how to act in an unknown environment solely by interaction. The agent's goal is to find a policy for selecting actions that maximises its expected utility. More specifically, we consider a discrete-time setting, such that at time t the agent observes a reward $r_t \in \mathbb{R}$, while its utility is the random quantity:

$$U = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1.1}$$

where $\gamma \in (0, 1)$ is a discount factor. The expected utility of policy $\pi \in \mathit{\Pi}$ for environment $\mu \in \mathcal{M}$ is denoted by $\mathbb{E}_{\mu,\pi} U$. However, since the environment μ is unknown to the agent, optimising with respect to π is not possible.

We consider reinforcement learning problems where the underlying environment is a Markov decision process (MDP). In this setting at each time step t the agent observes the environment state $s_t \in \mathcal{S}$, as well as the reward r_t. The agent then takes an action $a_t \in \mathcal{A}$, before observing a new state-reward pair (s_{t+1}, r_{t+1}). In MDPs, the environment dynamics are Markovian. Consequently, for a given MDP $\mu \in \mathcal{M}$ we have:

$$\mathbb{P}_\mu(s_{t+1} \in S \mid s_t = s, a_t = a) = \mathcal{T}_\mu(S \mid s, a) \tag{1.2}$$

where $\mathcal{T}_\mu$ is a conditional measure on the space of states $\mathcal{S}$. That is, $\mathcal{T}_\mu(S \mid s, a)$ is the probability that the next state is in the set S when we take action a from state s in the MDP μ. We assume that the reward is a deterministic function of the state and action $r_{t+1} = \mathcal{R}_\mu(s_t, a_t)$.

This paper focuses on Bayesian methods for solving the reinforcement learning problem (see [Vlassis *et al.*, 2012] for an overview). This is a decision-theoretic approach [DeGroot, 1970], with two key ideas. The first is to select an appropriate prior distribution ξ about the unknown environment, such that $\xi(\mu)$ represents our subjective belief that μ is the true environment. The second is to replace the expected utility over the real environment, which is unknown, with the expected utility under the subjective belief ξ, i.e.

$$\mathbb{E}_{\xi,\pi} U = \int_{\mathcal{M}} (\mathbb{E}_{\mu,\pi} U) \, \mathrm{d}\xi(\mu). \tag{1.3}$$

Formally, it is then possible to optimise with respect to the policy which maximises the expected utility over all possible environments, according to our belief. However, our future observations will alter our future beliefs according to Bayes' theorem. In particular the posterior mass placed on a set of MDPs $B \subset \mathcal{M}$ given a history h_t composed of a sequence of states, $s^t = s_1, \ldots, s_t$, actions $a^{t-1} = a_1, \ldots, a_{t-1}$, is:

$$\xi(B \mid h_t) \triangleq \frac{\int_B \prod_{k=1}^t \frac{\mathrm{d}}{\mathrm{d}\nu} \mathcal{T}_\mu(s_{t+1} \mid a_t, s_t) \, \mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \prod_{k=1}^t \frac{\mathrm{d}}{\mathrm{d}\nu} \mathcal{T}_\mu(s_{t+1} \mid a_t, s_t) \, \mathrm{d}\xi(\mu)}, \tag{1.4}$$

where $\frac{\mathrm{d}}{\mathrm{d}\nu} \mathcal{T}_\mu$ denotes the Radon-Nikodym derivative with respect to some measure ν on $\mathcal{S}$.[1] Consequently, the Bayes-optimal policy must take into account all potential future belief changes. For that reason, it will not in general be Markovian with respect to the states, but will depend on the complete history.

Most previous work on Bayesian reinforcement learning in continuous environments has focused on Gaussian process models for estimation. However, these suffer from two limitation. Firstly, they have significant computational complexity. Secondly, each dimension of the predicted state distribution is modeled independently. In this paper, we investigate the use of Bayesian inference under the assumption that the dynamics are (perhaps under a transformation) linear. Then the modeling problem becomes multivariate Bayesian linear regression, for which we can calculate (1.4) efficiently online.

[1] In the discrete case we may simply use $\mathbb{P}_\mu(s_{t+1} \mid a_t, s_t)$.

An other novelty of our approach in this context is that we do not simply use the common heuristic of acting as though the most likely or expected model is correct. Instead, generate a *sample model* from the posterior distribution. We then draw trajectories from the sampled model and collect simulated data which we use to obtain a policy. The policy is then executed in the real environment. This form of *Thompson sampling* is known to be a very efficient exploration method in bandit and discrete problems. We also show its efficacy for continuous domains.

The remainder of this paper is organised as follows. Section 2 gives an overview of related work and our contribution. Section 3 formally introduces our approach, with a description of the inference model in Sec. 3.1, and the policy selection method used in Sec. 3.2. Finally, the details of the online and off-line versions of our algorithms are detailed in Sec. 3.3. Experimental results are presented in Sec. 4 and we conclude with a discussion of future directions in Sec. 5.

2 Related work and our contribution

As mentioned in the introduction, Bayesian reinforcement learning models the reinforcement learning problem as a decision-theoretic problem by placing a prior distribution ξ on the set of possible MDPs $\mathcal{M}$. However, the exact solution of the decision problem is generally intractable as the transformed problem becomes a Markov decision processes with exponentially many states [Duff, 2002].

One of the first and most interesting approaches for approximate Bayesian reinforcement learning is *Thompson sampling*, which is also used in this paper. The idea is to sample a model from the posterior distribution, calculate the optimal policy for the sampled model, and then follow it for some period [Strens, 2000]. Thompson sampling has been recently shown to perform very well both in theory and practice in bandit problems [Kaufmanna *et al.*, 2012; Agrawal and Goyal, 2012]. Extensions and related models include Bayesian sparse sampling [Wang *et al.*, 2005], which uses Thompson sampling to deal with node expansion in the tree search. Taking multiple samples from the posterior can be used to estimate upper and lower bounds on the Bayes-optimal value function, which can then be used for tree search algorithms [Dimitrakakis, 2008; 2010]. Multiple samples can also be used to create an augmented optimistic model [Asmuth *et al.*, 2009; Castro and Precup, 2010]; or they can be used to construct a better lower bound [Dimitrakakis, 2011]. Finally, multiple samples can also be combined via voting schemes [Doshi-Velez, 2009]. Other approaches attempt to build optimistic models without sampling. For example [Kolter and Ng, 2009] adds an exploration bonus to rewards, while [Araya *et al.*, 2012] uses optimistic transition functions by constructing an augmented MDP in a Bayesian analogue of UCRL [Jacksh *et al.*, 2010].

For continuous state spaces, most Bayesian approaches have focused on Gaussian process (GP) models [Rasmussen and Kuss, 2004; Jung and Stone, 2010; Engel *et al.*, 2005; Reisinger *et al.*, 2008; Deisenroth *et al.*, 2009]. There are two key ideas that set our method apart from this work. Firstly, GP models are typically employed independently for each state feature. In contrast, the model we use deals naturally with correlated state features – consequently, less data may be necessary. Secondly, we do not calculate policies using the expected transition dynamics of the environment, as this is known to have potentially bad effects [Poupart *et al.*, 2006; Dimitrakakis, 2011]. Instead, value functions and policies are calculated by *sampling* from the posterior distribution of environments. This also important for efficient exploration.

This paper proposes a linear model-based Bayesian framework for reinforcement learning, for arbitrary state spaces $\mathcal{S}$ and for discrete action spaces $\mathcal{A}$ using Thompson sampling. First, we define a prior distribution on *linear dynamical models*, using a suitably chosen *basis*. Bayesian inference in this model is *fully closed form*, so that given a set of example trajectories it is easy to sample a model from the posterior distribution. For each such sample, we estimate the optimal policy. Since closed-form calculation of the optimal policy is not possible for general cost functions even with linear dynamics, we use approximate dynamic programming (ADP, see [Bertsekas, 2005] for an overview) with trajectories drawn from the sampled model. The resulting policy can then be applied to the real environment.

We experimented with two different ADP approaches for finding a policy for a given sampled MDP. Both are approximate policy iteration (API) schemes, using a set of trajectories generated from the sampled MDP to estimate a sequence of value functions and policies. For the policy evaluation step, we experimented with fitted value iteration (FVI) [Ernst *et al.*, 2005] and least-square temporal differences (LSTD) [Bradtke and Barto, 1996].

In the case where we use LSTD, the approach can be seen as an online, Bayesian generalisation of least-squares policy iteration (LSPI) [Lagoudakis and Parr, 2003]. Instead of performing LSTDQ on the empirical transition matrix, we perform a least-squares fit on a sample model drawn from the posterior. This fit can be very accurate by drawing a large amount of simulated trajectories in the sampled model.

We consider two applications of this approach. In the *offline* case, data is collected using a uniformly random policy. We then generate a model from the posterior distribution and calculate a policy for it, which is then evaluated in the real environment. In the *online* case, data is collected using policies generated from the sampled models, as in Thompson sampling. At the beginning each episode, a model is sampled from the posterior and the resulting policy is executed in the real environment. Thus, there is no separate data collection and evaluation phase. Our results show that this approach successfully finds optimal policies quickly and consistently both in the online and in the offline case, and that it has a significant overall advantage over LSPI.

3 Linear Bayesian reinforcement learning

The model presented in this paper uses Bayesian inference to estimate the environment dynamics. The assumption is that, with a suitable basis, these dynamics are linear with Gaussian noise. Unlike approaches using Gaussian processes, however, the next-state distribution is not modeled using a product distribution, i.e. we do not assume that

the various components of the state are independent. In a further innovation, rather than using the expected posterior parameters, we employ sampling from the posterior distribution. For each sampled model, we then obtain an approximately optimal policy by using approximate dynamic programming, which is then executed in the real environment. This form of *Thompson sampling* [Strens, 2000; Thompson, 1933] allows us to perform efficient exploration, with the policies naturally becoming greedier as the posterior distribution converges.

3.1 The predictive model

In our model we assume that, for a state set $\mathcal{S}$ there exists a mapping $f : \mathcal{S} \to \mathcal{X}$ to a k-dimensional vector space $\mathcal{X}$ such that the transformed state at time t is $\boldsymbol{x}_t \triangleq f(\boldsymbol{s}_t)$. The next state $\boldsymbol{s}_{t+1}$ is given by the output of a function $g : \mathcal{X} \times \mathcal{A} \to \mathcal{S}$ of the transformed state, the action and some additive noise:

$$\boldsymbol{s}_{t+1} = g(\boldsymbol{x}_t, a_t) + \varepsilon_t. \tag{3.1}$$

In this paper, we model the noise ε_t and the function g as a multivariate linear-Gaussian model. This is parameterized via a set of $k \times k$ *design* matrices $\{ \boldsymbol{A}_i \mid i \in \mathcal{A} \}$, such that $g(\boldsymbol{x}_t, a_t) = \boldsymbol{A}_{a_t} \boldsymbol{x}_t$ and a set of *covariance* matrices $\{ \boldsymbol{V}_i \mid i \in \mathcal{A} \}$ for the noise. Then, the next state distribution is:

$$\boldsymbol{s}_{t+1} \mid \boldsymbol{x}_t = \boldsymbol{x}, a_t = i \sim \mathcal{N}(\boldsymbol{A}_i \boldsymbol{x}, \boldsymbol{V}_i). \tag{3.2}$$

In order to model our uncertainty with a (subjective) prior distribution ξ, we have to specify the model structure. In our model, we do not assume independence between the output dimensions, something which could potentially make inference difficult. Fortunately, in this particular case, a conjugate prior exists in the form of the *matrix-normal distribution* for $\boldsymbol{A}$ and the *inverse-Wishart* distribution for $\boldsymbol{V}$. Given $\boldsymbol{V}_i$, the distribution for $\boldsymbol{A}_i$ is matrix-normal, while the marginal distribution of $\boldsymbol{V}_i$ is inverse-Wishart. More specifically,

$$\boldsymbol{A}_i \mid \boldsymbol{V}_i = \widehat{\boldsymbol{V}} \sim \phi(\boldsymbol{A}_i \mid \boldsymbol{M}, \boldsymbol{C}, \widehat{\boldsymbol{V}}) \tag{3.3}$$

$$\boldsymbol{V}_i \sim \psi(\boldsymbol{V}_i \mid \boldsymbol{W}, n), \tag{3.4}$$

where ϕ_i is the prior distribution on dynamics matrices conditional on the covariance and two prior parameters: $\boldsymbol{M}$, which is the prior mean and $\boldsymbol{C}$ which is the prior output (dependent variable) covariance. Finally, ψ is the marginal prior on covariance matrices, which has an inverse-Wishart distribution with $\boldsymbol{W}$ and n. More precisely, the distributions are:

$$\phi(\boldsymbol{A}_i \mid \boldsymbol{M}, \boldsymbol{C}, \widehat{\boldsymbol{V}}) \propto e^{-\frac{1}{2} \operatorname{tr}\left[(\boldsymbol{A}_i - \boldsymbol{M})^\top \boldsymbol{V}_i^{-1} (\boldsymbol{A}_i - \boldsymbol{M}) \boldsymbol{C}\right]}, \tag{3.5}$$

$$\psi(\boldsymbol{V}_i \mid \boldsymbol{W}, n) \propto |\boldsymbol{V}^{-1} \boldsymbol{W} / 2|^{n/2} e^{-\frac{1}{2} \operatorname{tr}(\boldsymbol{V}^{-1} \boldsymbol{W})}. \tag{3.6}$$

Essentially, the model is an extension of the univariate Bayesian linear regression model (see for example [DeGroot, 1970]) to the multivariate case via vectorisation of the mean matrix. Since the prior is conjugate, it is relatively simple to calculate posterior values of the parameters after each observation. While we omit the details, a full description of inference using this model is given in [Minka, 2001].

Throughout this text, we shall employ $\xi_t = (\phi_t, \psi_t)$ to denote our posterior distributions at time t, with ξ_t referring to our complete posterior. The remaining problem is how to estimate the Bayes-expected utility of the current policy and how to perform policy improvement.

3.2 Policy evaluation and optimisation

In the Bayesian setting, policy evaluation and optimisation are not trivial. The most common method used is the expected MDP heuristic, where policies are evaluated or optimised on the expected MDP. However, this ignores the shape of the posterior distribution. Alternatively, policies can be evaluated via Monte Carlo sampling, but then optimisation becomes hard. A good heuristic that does not ignore the complete posterior distribution and for which it is easy to calculate a policy, called Thompson sampling, is the one we shall actually employ in this paper. The following paragraphs give a quick overview of each method.

Expected MDP A naive way to estimate the expected utility of a policy is to first calculate the expected (or most probable) dynamics, and then use either an exact or an approximate dynamic programming algorithm. This may very well be a good idea if the posterior distribution is sharply concentrated around the mean, since then:

$$\mathbb{E}_{\xi,\pi} U \approx \mathbb{E}_{\mu_\xi,\pi} U, \qquad \mu_\xi \triangleq \mathbb{E}_\xi \mu. \tag{3.7}$$

where μ_ξ is the expected MDP model.[2] However, as pointed out in [Araya *et al.*, 2012; Dimitrakakis, 2011] this approach may give completely incorrect results,

Monte Carlo sampling An alternative method is to take a number of samples μ_i from the current posterior distribution and then calculate the expected utility of each, i.e.

$$\mathbb{E}_{\xi,\pi} U = \frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{\mu_i,\pi} U + O(K^{-1/2}), \quad \mu_i \sim \xi_t. \tag{3.8}$$

This form of Monte Carlo sampling gives much more accurate results, at the expense of some additional computation. However, finding an optimal policy over a set of sampled MDPs is difficult even for restricted classes of policies [Dimitrakakis, 2011]. Nevertheless, Monte Carlo sampling can also be used to obtain stochastic upper and lower bounds on the value function, which can be used to improve the policy search [Dimitrakakis, 2010; 2008].

Thompson sampling An interesting special case is when we only sample a single MDP, i.e. when we perform Monte Carlo sampling with $K = 1$. Then it is relatively easy to calculate the optimal policy for this sample. This method, which we employ in this work, is called *Thompson sampling*, and was first used in the context of reinforcement learning by [Strens, 2000]. The idea is to sample an MDP from the current posterior and then calculate a policy that is optimal with respect to that MDP. We then execute this policy in the environment. The major advantage of Thompson sampling is that it is known to result in a very efficient form of exploration (see for example [Agrawal and Goyal, 2012] for recent results on bandit problems).

[2] Similar problems exist when using the most probable MDP instead.

3.3 Algorithm overview

We can now put everything together for the complete linear Bayesian reinforcement learning (LBRL) algorithm. The algorithm has four steps. Firstly, sampling a model from the posterior distribution. Secondly, using the sampled model to calculate a new policy. Finally, executing this policy in the real environment. In the online version of the algorithm the data obtained by executing this policy is then used to calculate a new posterior distribution.

Sampling from the posterior Our posterior distribution at time t is $\xi_t = (\phi_t, \psi_t)$, with ψ_t being the marginal posterior on covariance matrices, and ϕ_t being the posterior on design matrices (conditional on the covariance). In order to sample a model from the posterior, we first draw a covariance matrix $\boldsymbol{V}_i$ using (3.4) for every action $i \in \mathcal{A}$, and then plug those into (3.3) to generate a set of design matrices $\boldsymbol{A}_i$. The first step requires sampling from the inverse-Wishart distribution (which can be done efficiently using the algorithm suggested by [Smith and Hocking, 1972]), and the second from the matrix-normal distribution.

ADP on the sampled MDP Given an MDP μ sampled from our posterior belief, we can calculate a nearly-optimal policy π using approximate dynamic programming (ADP) on μ. This can be done with a number of algorithms. Herein, we investigated two approximate policy iteration (API) schemes, using either fitted value iteration (FVI) or least-squares temporal differences (LSTD) for the policy evaluation step. Both of these algorithms require sample trajectories from the environment. This is fortunately very easy to achieve, since we can use the sampled model to generate any number of trajectories arbitrarily. Consequently, we can always have enough *simulated* data to perform a good fit with FVI or LSTD.[3] We note here that API using LSTD additionally requires a generative model for the policy improvement step. Happily, we can use the sampled MDP μ for that purpose.[4]

Algorithm 1 LBRL: Linear Bayesian reinforcement learning

Input Basis f, ADP parameters P, prior ξ_0
for episode k **do**
  $\mu^{(k)} \sim \xi_{t_k}(\mu)$ // generate MDP from posterior
  $\pi^{(k)} = \texttt{ADP}(\mu^{(k)}, P)$ // Get new policy
  for $t = t_k, \ldots, t_{k+1} - 1$ **do**
    $a_t \mid s_t = s \sim \pi^{(k)}(a \mid s)$ // Take action
    $\xi_{t+1}(\mu) = \xi_t(\mu \mid s_{t+1}, a_t, s_t)$ // Update posterior
  end for
end for

Offline LBRL In the offline version of the algorithm, we simply collect a set of trajectories from a *uniformly random policy*, comprising a history h_t of length t. Then, we sample an MDP from the posterior $\xi_t(\mu) = \xi_0(\mu \mid h_t)$ and calculate the optimal policy for the sample using ADP. This policy is then *evaluated* on the real environment.

Online LBRL In the online version of the algorithm, shown in Alg. 1 we collect samples using our *own generated policies*. We begin with some initial belief $\xi_0 = (\phi_0, \psi_0)$ and a uniformly random policy $\pi^{(0)}$. This policy is executed until either the episode ends naturally or due to reaching a time-limit T. At the k-th episode, which starts at time t_k, we sample a new MDP $\mu^{(k)} \sim \xi_{t_k}$ from our current posterior $\xi_{t_k}(\cdot) = \xi(\cdot \mid h_{t_k})$ and then calculate a near-optimal stationary policy for $\mu^{(k)}$:

$$\pi^{(k)} \approx \arg\max_{\pi} \mathbb{E}_{\mu^{(k)}, \pi} U,$$

such that $\pi^{(k)}(a \mid s)$ is a conditional distribution on actions $a \in \mathcal{A}$ given states $s \in \mathcal{S}$. This policy is then executed in the real environment until the end of the episode and the data collected are used to calculate the new posterior. As the calculation of posterior parameters is fully incremental, we incur no additional computational cost for running this algorithm online.

4 Experiments

We conducted two sets of experiments to analyze both the offline and the online performance of the various algorithms. Comparisons have been made with the well-known least square policy iteration (LSPI) algorithm [Lagoudakis and Parr, 2003] for the offline case, as well as an online variant [Buşoniu *et al.*, 2010] for the online case. We used preliminary runs and guidance from the literature to select the features for the LSTDQ algorithm used in the inner loop of LSPI. The source for all the experiments can be found in [Dimitrakakis *et al.*,].

We employed the same features for the ADP algorithms used in LBRL. However, the basis used for the Bayesian regression model in LBRL was simply $f(\boldsymbol{s}) \triangleq [\boldsymbol{s}, 1]^\top$. In preliminary experiments, we found this sufficient for a high-quality approximation. After that, we use API to find a good policy for a sampled MDP, where we experimented with regularised FVI and LSTD for the policy evaluation step, adding a regularisation factor $10^{-2}\boldsymbol{I}$. In both cases, we drew single step transitions from a set of 3000 uniformly drawn states from the sampled model.

For the offline performance evaluation, we first drew rollouts from $k = \{50, 100, \ldots, 1000\}$ states drawn from the environment's starting distribution, using a uniformly random policy. The maximum horizon of each rollout was set equal to 40. The collected data was then fed to each algorithm in order to produce a policy. This policy was evaluated over 1000 rollouts on the environment.

In the online case, we simply use the last policy calculated by each algorithm at the end of the last episode, so there is no separate learning and evaluation phase. This means that efficient exploration must be performed. For LBRL, this is done using Thompson sampling. For online-LSPI, we followed the

[3] These use no data collected in the real environment.

[4] In preliminary experiments, we also investigated the use of fitted Q-iteration and LSPI, but found that these had inferior performance.

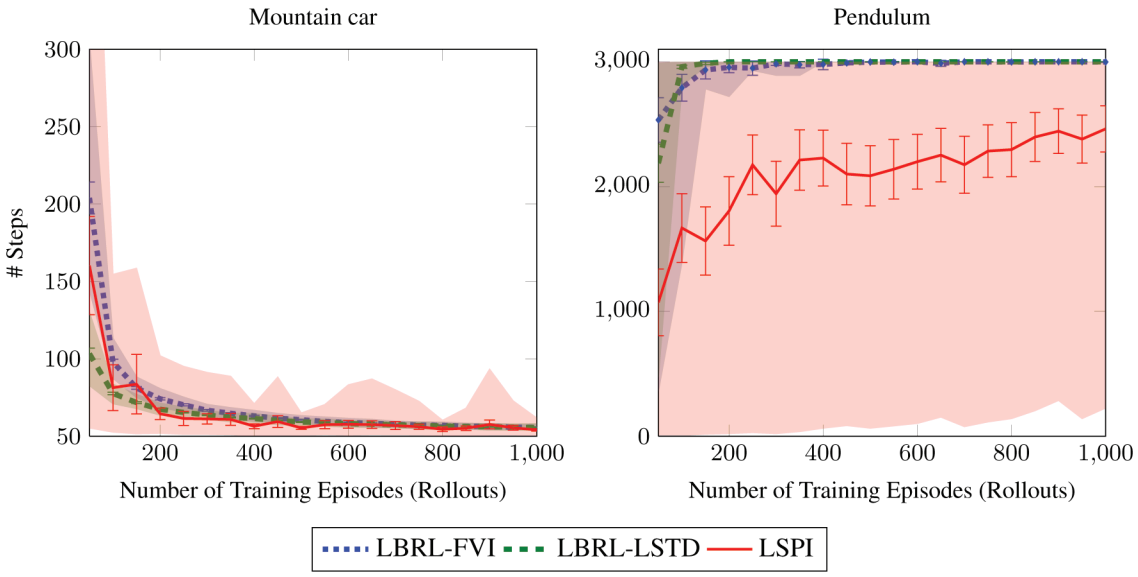

Figure 1: Offline performance comparison between LBRL-FVI, LBRL-LSTD and LSPI. The error bars show 95% confidence intervals, while the shaded regions show 90% percentiles over 100 runs.

approach of [Buşoniu *et al.*, 2010], who adopts an ϵ-greedy exploration scheme with an exponentially decaying schedule $\epsilon_t = \epsilon_d^t$, with $\epsilon_0 = 1$. In preliminary experiments, we found $\epsilon_d = 0.9968$ to be a reasonable compromise. We compared the algorithms online for 1000 episodes.

4.1 Inverted pendulum

The first set of experiments includes the *inverted pendulum* domain, which tries to balance a pendulum by applying forces of a mixed magnitude (50 Newtons). The state space consists of two continuous variables, the vertical angle (θ) and the angular velocity ($\dot{\theta}$) of the pendulum. The agent has at his arsenal three actions: no force, left force or right force. A zero reward is received at each time step except in the case where the pendulum falls ($|\theta| \leq \pi/2$). In this case, a negative (-1) reward is given and a new rollout begins. Each rollout starts by setting the pendulum in a perturbed state close to the equilibrium point. More information about the environment dynamics can be found at [Lagoudakis and Parr, 2003]. Each rollout is allowed to run for 3000 steps at maximum. Additionally, the discount factor is set to 0.95. For FVI/LSTD and LSPI, we used an equidistant 3×3 grid of RBFs over the state space following the suggestions of [Lagoudakis and Parr, 2003], which was replicated for each action for the LSTDQ algorithm used in LSPI.

4.2 Mountain car

In the second experimental set, we have used the *mountain car* environment. Two continuous variables characterise the vehicle state in the domain, its position (p) and its velocity (u). The objective in this task is to drive an underpowered vehicle up a steep road from a randomly selected position to the right hilltop ($p \geq 0.5$) with at most 1000 steps. In order to achieve our goal, we can select between three actions: forward, reverse and zero throttle. The received reward is -1 except in the case where the target is reached (zero reward). At the beginning of each rollout, the vehicle is positioned to a new state, with the position and the velocity uniformly randomly selected. The discount factor is equal to 0.999. An equidistant 4×4 grid of RBFs over the state space plus a constant term is selected for FVI/LSTD and LSPI.

4.3 Results

In our results, we show the average performance in terms of number of steps of each method, averaged over 100 runs. For each average, we also plot the 95% confidence interval for the accuracy of the mean estimate with error bars. In addition, we show the 90% percentile region of the runs, in order to indicate inter-run variability in performance.

Figure 1 shows the results of the experiments in the *offline* case. For the *mountain car*, it is clear that the most stable approach is LBRL-LSTD, while LSPI is the most unstable. Nevertheless, on average the performance of LBRL-LSTD and LSPI is similar, while LBRL-FVI is slightly worse. For the *pendulum* domain, the performance of LBRL remains quite good, with LBRL-LSTD being the most stable. While LSPI manages to find the optimal policy frequently, nevertheless around 5% of its runs fail.[5]

Figure 2 shows the results of the experiments in the *online* case. For the *mountain car*, both LSPI and LBRL-LSTD managed to find an excellent policy in the vast majority of runs. In the *pendulum* domain, we see that LBRL-LSTD significantly outperforms LSPI. In particular, after 80 episodes all more than 90% of the runs are optimal, while

[5]We note that the results presented in [Lagoudakis and Parr, 2003] for LSPI are slightly better, but remain significantly below the LBRL results.

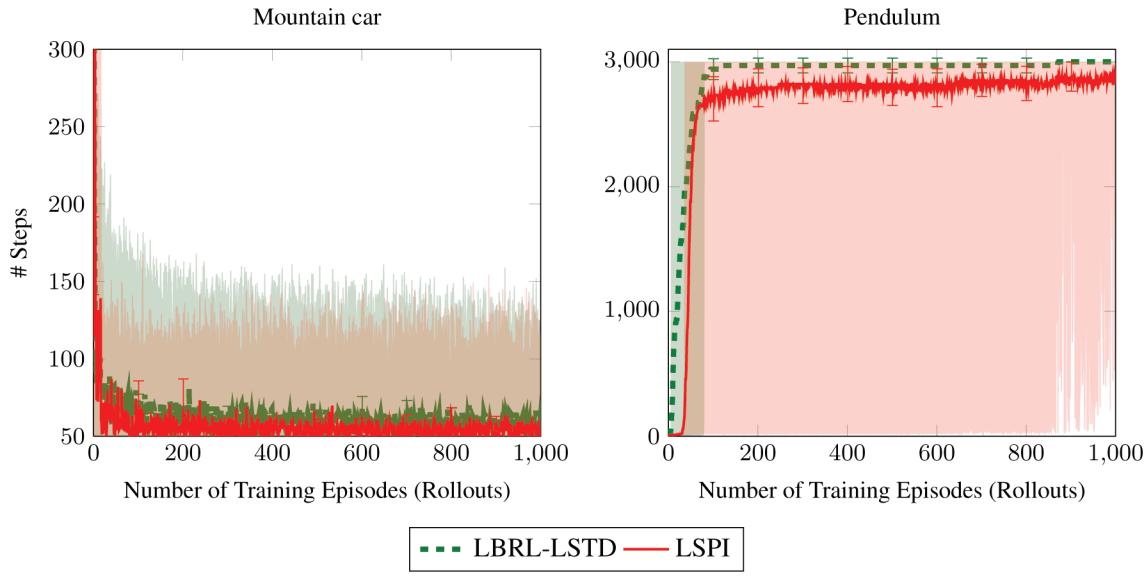

Figure 2: Online performance comparison between LBRL-LSTD and LSPI. The error bars show 95% confidence intervals, while the shaded regions show 90% percentiles over 100 runs.

many LSPI runs fail to find a good solution even after hundreds of episode. The mean difference is somewhat less spectacular, though still significant.

The success of LBRL-LSTD over LSPI can be attributed to a number of reasons. Firstly, it could be the *more efficient exploration*. Indeed, in the mountain car domain, where the starting state distribution is uniform, we can see that LBRL and LSPI have very similar performance. Another possible reason is that LBRL also makes *better use of the data*, since it uses it to calculate the posterior distribution over MDP dynamics. It is then possible to perform very accurate ADP using simulated data from a model sampled from the posterior. This is supported by the offline results in the pendulum domain.

5 Conclusion

We presented a simple linear Bayesian approach to reinforcement learning in continuous domains. Unlike Gaussian process models, by using a linear-Gaussian model, we have the potential to scale up to real world problems which Bayesian reinforcement learning usually fails to solve with a reasonable amount of data. In addition, this model easily takes into account correlations in the state features, further reducing sample complexity. We solve the problem of computing a good policy in continuous domains with uncertain dynamics by using Thompson sampling. This not much more expensive than computing the expected MDP and forces a natural exploration behaviour.

In practice, the algorithm is at least as good as LSPI in offline mode, while being considerably more stable overall. When LBRL is used to perform online exploration, we find that the algorithm very quickly converges to a near-optimal policy and is extremely stable. Experimentally, it would be interesting to compare LBRL with standard GP methods that employ the expected MDP heuristic.

Thompson sampling could be used with other Bayesian models for continuous state spaces. A natural extension would thus be to move to a non-parametric model, e.g. replace the multivariate linear model with a multivariate Gaussian process. The major hurdle would be the computational cost. Consequently, in future work we would like to use a recently proposed methods for efficient Gaussian processes in the multivariate case, such as [Alvarez *et al.*, 2011] that uses convolution processes. Other Bayesian schemes for multivariate regression analysis [Mehmet, 2012] may be applicable as well.

Finally, it would be highly interesting to consider other exploration methods. One example is the Monte-Carlo extension of Thompson sampling used in [Dimitrakakis, 2011], which can also be used for continuous state spaces. Other approaches, such as the optimistic transition MDP used in [Araya *et al.*, 2012] may not be so straightforward to adopt to the continuous case. Nevertheless, while these approaches may be costly computationally, we believe that they will be beneficial in terms of performance.

Acknowledgements

We wish to thank the anonymous reviewers for their excellent comments and suggestions. This work was partially supported by the Marie Curie Project ESDEMUU, Grant Number 237816 and by an ERASMUS exchange grant.

References

[Agrawal and Goyal, 2012] S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *COLT 2012*, 2012.

[Alvarez *et al.*, 2011] M. Alvarez, D. Luengo-Garcia, M. Titsias, and N. Lawrence. Efficient multioutput gaussian processes through variational inducing kernels. 2011.

[Araya *et al.*, 2012] M. Araya, V. Thomas, O. Buffet, et al. Near-optimal BRL using optimistic local transitions. In *ICML*, 2012.

[Asmuth *et al.*, 2009] J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *UAI 2009*, 2009.

[Bertsekas, 2005] D. Bertsekas. Dynamic programming and suboptimal control: From ADP to MPC. *Fundamental Issues in Control, European Journal of Control*, 11(4-5), 2005. From 2005 CDC, Seville, Spain.

[Bradtke and Barto, 1996] S.J. Bradtke and A.G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996.

[Buşoniu *et al.*, 2010] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Online least-squares policy iteration for reinforcement learning control. In *Proceedings of the 2010 American Control Conference*, pages 486–491, 2010.

[Castro and Precup, 2010] P. Castro and D. Precup. Smarter sampling in model-based Bayesian reinforcement learning. *Machine Learning and Knowledge Discovery in Databases*, pages 200–214, 2010.

[DeGroot, 1970] M. H. DeGroot. *Optimal Statistical Decisions*. John Wiley & Sons, 1970.

[Deisenroth *et al.*, 2009] M.P. Deisenroth, C.E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

[Dimitrakakis *et al.*,] C. Dimitrakakis, N. Tziortziotis, and A. Tossou. Beliefbox: A framework for statistical methods in sequential decision making. http://code.google.com/p/beliefbox/.

[Dimitrakakis, 2008] C. Dimitrakakis. Tree exploration for Bayesian RL exploration. In *Computational Intelligence for Modelling, Control and Automation, International Conference on*, pages 1029–1034, Wien, Austria, 2008. IEEE Computer Society.

[Dimitrakakis, 2010] C. Dimitrakakis. Complexity of stochastic branch and bound methods for belief tree search in Bayesian reinforcement learning. In *ICAART 2010*, pages 259–264. Springer, 2010.

[Dimitrakakis, 2011] C. Dimitrakakis. Robust bayesian reinforcement learning through tight lower bounds. In *European Workshop on Reinforcement Learning (EWRL 2011)*, number 7188 in LNCS, pages 177–188, 2011.

[Doshi-Velez, 2009] Finale Doshi-Velez. The infinite partially observable Markov decision process. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2009. MIT Press.

[Duff, 2002] M. O. Duff. *Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

[Engel *et al.*, 2005] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian process. In *International Conference on Machine Learning*, pages 201–208, 2005.

[Ernst *et al.*, 2005] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[Jacksh *et al.*, 2010] T. Jacksh, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

[Jung and Stone, 2010] T. Jung and P. Stone. Gaussian processes for sample-efficient reinforcement learning with RMAX-like exploration. In *ECML/PKDD 2010*, pages 601–616, 2010.

[Kaufmanna *et al.*, 2012] E. Kaufmanna, N. Korda, and R. Munos. Thompson sampling: An optimal finite time analysis. In *ALT-2012*, 2012.

[Kolter and Ng, 2009] J. Z. Kolter and A. Y. Ng. Near-Bayesian exploration in polynomial time. In *ICML 2009*, 2009.

[Lagoudakis and Parr, 2003] M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[Mehmet, 2012] G. Mehmet. A bayesian multiple kernel learning framework for single and multiple output regression. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pages 354–359, 2012.

[Minka, 2001] T. P. Minka. Bayesian linear regression. Technical report, Microsoft research, 2001.

[Poupart *et al.*, 2006] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *ICML 2006*, pages 697–704. ACM Press New York, NY, USA, 2006.

[Rasmussen and Kuss, 2004] C.E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 751–759, 2004.

[Reisinger *et al.*, 2008] J. Reisinger, P. Stone, and R. Miikkulainen. Online kernel selection for bayesian reinforcement learning. In *International Conference on Machine Learning*, pages 816–823, 2008.

[Smith and Hocking, 1972] WB Smith and RR Hocking. Wishart variates generator, algorithm as 53. *Applied Statistics*, 21:341–345, 1972.

[Strens, 2000] M. Strens. A Bayesian framework for reinforcement learning. In *ICML 2000*, pages 943–950, 2000.

[Thompson, 1933] W.R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of two Samples. *Biometrika*, 25(3-4):285–294, 1933.

[Vlassis *et al.*, 2012] N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart. *Reinforcement Learning*, chapter Bayesian Reinforcement Learning, pages 359–386. Springer, 2012.

[Wang *et al.*, 2005] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML '05*, pages 956–963, New York, NY, USA, 2005. ACM.