ABC Reinforcement Learning

Christos Dimitrakakis CHRISTOS.DIMITRAKAKIS@GMAIL.COM
EPFL, Lausanne, Switzerland

Nikolaos Tziortziotis NTZIORZI@GMAIL.COM
University of Ioannina, Greece

Abstract

We introduce a simple, general framework for *likelihood-free* Bayesian reinforcement learning, through Approximate Bayesian Computation (ABC). The advantage is that we only require a prior distribution on a class of simulators. This is useful when a probabilistic model of the underlying process is too complex to formulate, but where detailed simulation models are available. ABC-RL allows the use of any Bayesian reinforcement learning technique in this case. It can be seen as an extension of simulation methods to both planning and inference. We experimentally demonstrate the potential of this approach in a comparison with LSPI. Finally, we introduce a theorem showing that ABC is sound.

1. Introduction

Bayesian reinforcement learning (Strens, 2000; Vlassis et al., 2012) is the decision-theoretic approach (DeGroot, 1970) to solving the reinforcement learning problem. However, apart from the fact that calculating posterior distributions and the Bayes-optimal decision is frequently intractable (Duff, 2002; Ross et al., 2008), another major difficulty is the specification of the prior and model class. While there exist a number of non-parametric Bayesian model classes which can be brought to bear for estimation of the dynamics of an unknown process, it may not be a trivial matter to select the correct class and prior. On the other hand, it is frequently known that the process can be approximated well by a complex parametrised simulator. The question is how to take advantage of this knowledge when the best simulator parameters are not known.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

We propose a simple, general, reinforcement learning framework employing the principles of Approximate Bayesian Computation (ABC, see (Csilléry et al., 2010) for an overview) for performing Bayesian inference using simulation. In doing so, we extend rollout algorithms for reinforcement learning, such as those described in (Bertsekas, 2006; Bertsekas & Tsitsiklis, 1996; Dimitrakakis & Lagoudakis, 2008; Lagoudakis & Parr, 2003a), to the case where we do not know what the correct model to draw rollouts from is.

We show how to use ABC to compute approximate posteriors over a set of environment models in the context of reinforcement learning. This includes a simple but general theoretical result on the quality of ABC posterior approximations. Finally, building on previous approaches to Bayesian reinforcement learning, we propose a strategy for selecting policies in this setting.

1.1. The setting

In the *reinforcement learning problem*, an agent is acting in some unknown environment μ, according to some policy π. The agent's policy is a procedure for selecting a sequence of actions, with the action at time t being $a_t \in \mathcal{A}$. The environment reacts to this sequence with a corresponding sequence of observations $x_t \in \mathcal{X}$ and rewards $r_t \in \mathbb{R}$. This interaction may depend on the complete history[1] $h \in \mathcal{H}$, where $\mathcal{H} \triangleq (\mathcal{X} \times \mathcal{A} \times \mathbb{R})^*$ is the set of all state action reward sequences, as neither the agent or the environment are necessarily finite-order Markov. For example, the agent may learn, or the environment may be partially observable.

In this paper, we use a number of shorthands to simplify notation. Firstly, we denote the (random) probability measure for the agent's action at time t by:

$$\pi_t(A) \triangleq \mathbb{P}^\pi(a_t \in A \mid x^t, r^t, a^{t-1}), \qquad (1.1)$$

[1] A history may include multiple trajectories in episodic environments.

where x^t is a shorthand for the sequence $(x_i)_{i=1}^t$; similarly, we use x_k^t for $(x_i)_{i=k}^t$. We denote the environment's response at time $t+1$ given the history at time t by:

$$\mu_t(B) \triangleq \mathbb{P}_\mu((x_{t+1}, r_{t+1}) \in B \mid x^t, r^t, a^t). \tag{1.2}$$

In a further simplification, we shall also use $\pi_t(a_t)$ for the probability (or density) of the action actually taken by the policy at time t, and similarly, $\mu_t(x_t)$ for the realised observation. Finally, we use $\mathbb{P}_\mu^\pi$ to denote joint distributions on action, observation and reward sequences under the environment μ and policy π.

The agent's goal is determined through its utility:

$$U \triangleq \sum_{t=1}^{\infty} \gamma^{t-1} r_t, \tag{1.3}$$

which is a discounted sum of the total instantaneous rewards obtained, with $\gamma \in [0, 1]$. Without loss of generality, we assume that $U \in [0, U_{\max}]$. The optimals policy maximises the expected utility $\mathbb{E}_\mu^\pi U$. As in the reinforcement learning problem the environment μ is *unknown*, this maximisation is ill-posed. Intuitively, we can increase the expected utility by either: (i) Trying to better estimate μ in order to perform the maximisation later (exploration), or (ii) Use a best-guess estimate of μ to obtain high rewards (exploitation).

In order to solve this trade-off, we can adopt a Bayesian viewpoint (DeGroot, 1970; Savage, 1972), where we consider a (potentially infinite) set of environment models $\mathcal{M}$. In particular, we select a *prior probability* measure ξ on $\mathcal{M}$. For an appropriate subset $B \subset \mathcal{M}$, the quantity $\xi(B)$ describes our initial belief that the correct model lies in B. We can now formulate the alternative goal of maximising the expected utility with respect to our prior:

$$\mathbb{E}_\xi^\pi U = \int_{\mathcal{M}} (\mathbb{E}_\mu^\pi U) \, \mathrm{d}\xi(\mu). \tag{1.4}$$

We can now formalise the problem as finding a policy $\pi_\xi^* \in \arg\max_\pi \mathbb{E}_\xi^\pi U$. Any such policy is *Bayes-optimal*, as it solves the exploration-exploitation problem with respect to our prior belief.

1.2. Related work and our contribution

The first difficulty when adopting a Bayesian approach to sequential decision making is that finding the policy maximising (1.4) is hard (Duff, 2002) even in restricted classes of policies (Dimitrakakis, 2011). On the other hand, simple heuristics such as Thompson sampling (Strens, 2000; Thompson, 1933) provide an efficient trade-off (Agrawal & Goyal, 2012; Kaufmanna et al., 2012) between exploration and exploitation. Alghough other heuristics exist (Araya et al., 2012; Castro & Precup, 2007; Kolter & Ng, 2009; Poupart et al., 2006; Strens, 2000), in this paper we focus on an approximate version of Thompson sampling for reasons of simplicity. The second difficulty is that in many interesting problems, the exact posterior calculation may be intractable, mainly due to partial observability (Poupart & Vlassis, 2008; Ross et al., 2008). Interestingly, an ABC approach would not suffer from this problem for reasons that will be made clear in the sequel.

The most fundamental difficulty in a Bayesian framework is specifying a generative model class: it is not always clear what is the best model to use for an application. However, frequently we have access to a class of *parametrised simulators* for the problem. Therefore, one reasonable approach is to find a good policy for a simulator in the class, and then apply it to the actual problem. Methods for finding good policies using simulation have been extensively studied before (Bertsekas, 2006; Bertsekas & Tsitsiklis, 1996; Dimitrakakis & Lagoudakis, 2008; Gabillon et al., 2011; Wu et al., 2010). However, in all those cases simulation was performed on a simulator with *fixed* parameters.

Approximate Bayesian Computation (ABC) (see Csilléry et al., 2010; Marin et al., 2011, for an overview) is a general framework for likelihood-free Bayesian inference via simulation. It has been developed because of the existence of applications, such as econometric modelling (e.g. Geweke, 1999), where detailed simulators were available, but no useful analytical probabilistic models. While ABC methods have also been used for inference in dynamical systems (e.g Toni et al., 2009), they have not yet been applied to the reinforcement learning problem.

This paper proposes to perform Bayesian reinforcement learning through ABC on an arbitrary class of parametrised simulators. As ABC has been widely used in applications characterised by large amounts of data and complex simulations with many unknown parameters, it may also scale well in reinforcement learning applications. The proposed methodology is generally applicable to arbitrary problems, including partially observable environments, continuous state spaces, and stochastic Markov games.

ABC Reinforcement Learning generalises methods previously developed for simulation-based approximation of optimal policies to the Bayesian case. While in the standard framework covered by Bertsekas (1999), a particular simulator of the environment is assumed to exist, via ABC we can relax this assumption. We only

need a class of parametrised simulators that contain one close to the real environment dynamics. Thus, the only remaining difficulty is computational complexity.

Finally, we provide a simple but general bound for ABC posterior computation. This bounds the KL divergence of the approximate posterior computed via ABC and the complete posterior distribution. As far as we know, this is a new and widely applicable result, although some other theoretical results using similar assumptions appear in (Jasra et al., 2010) and in (Dean & Singh, 2011) for hidden Markov models.

Section 2 introduces ABC inference for reinforcement learning, discusses its difference from standard Bayesian inference, and presents a theorem on the quality of the ABC approximation. Section 3 describes the ABC-RL framework and the ABC-LSPI algorithm for continuous state spaces. An experimental illustration is given in Sec. 4, followed by a discussion in Sec. 5. The appendix contains the collected proofs.

2. Approximate Bayesian Computation

Approximate Bayesian Computation encompasses a number of likelihood-free techniques where only an approximate posterior is calculated via simulation. We first discuss how standard Bayesian inference in reinforcement learning differs from ABC inference. We then introduce a theorem on the quality of the ABC approximation.

2.1. Bayesian inference for reinforcement learning

Imagine that the history $h \in \mathcal{H}$ has been generated from a process $\mu \in \mathcal{M}$ controlled with a history-dependent policy π, something which we denote as $h \sim \mathbb{P}_\mu^\pi$. Now consider a prior ξ on $\mathcal{M}$ with the property that $\xi(\cdot \mid \pi) = \xi(\cdot)$, i.e. that the prior is independent of the policy used. Then the posterior probability, given a history h generated by a policy π, that $\mu \in B$ can be written as: [2]

$$\xi(B \mid h, \pi) = \frac{\int_B \mathbb{P}_\mu^\pi(h) \, \mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \mathbb{P}_\mu^\pi(h) \, \mathrm{d}\xi(\mu)}. \tag{2.1}$$

Fortunately, the dependence on the policy can be removed, since the posterior is the same for all policies that put non-zero mass on the observed data:

Remark 2.1. *Let $h \sim \mathbb{P}_\mu^\pi$. Then $\forall \pi' \neq \pi$ such that $\mathbb{P}_\mu^{\pi'}(h) > 0$, $\xi(B \mid h, \pi) = \xi(B \mid h, \pi')$.*

Consequently, when calculating posteriors, the policy employed need not be considered, even when the process and policy depend on the complete history. However, in the ABC setting we do not have direct access to the probabilities μ_t, for the models μ in our model class $\mathcal{M}$. However, we can always generate observations from any model: $x_{t+1} \sim \mu_t$. This idea is used by ABC to calculate approximate posterior distributions.

2.2. ABC inference for reinforcement learning

The main idea of ABC is to approximate samples from the posterior distribution via simulation. We produce a sequence of sample models $\mu^{(k)}$ from the prior ξ, and then generate data $h^{(k)}$ from each. If the generated data is "sufficiently close" to the history h, then the k-th model is accepted as a sample from the posterior $\xi(\mu \mid h)$. More specifically, ABC requires that we define an approximately sufficient statistic $f : \mathcal{H} \to \mathcal{W}$ on some normed vector space $(\mathcal{W}, \|\cdot\|)$. If $\|f(h) - f(h^{(k)})\| \leq \varepsilon$ then $\mu^{(k)}$ is accepted as a sample from the posterior. Algorithm 1 gives the sampling method in detail for reinforcement learning. An important difference with the standard ABC posterior approximation, as well as exact inference, is the dependency on π.

Note that even though Remark 2.1 declares that the posterior is independent of the policy used, when using ABC this is no longer true. We must maintain the complete policy used until then to generate samples, otherwise there is no way to generate a sequence of observations.[3] Intuitively, the algorithm can basically be seen as generating rollouts from a number of simulators, sampled from our prior distribution. The

Algorithm 1 ABC-RL-Sample

input Prior ξ on $\mathcal{M}$, history $h \in \mathcal{H}$, threshold ε, statistic $f : \mathcal{H} \to \mathcal{W}$, policy π, maximum number of samples N_{sam}, stopping condition τ.
$\widehat{M} = \emptyset$.
for $k = 1, \ldots, N_{\text{sam}}$ **do**
  $\mu^{(k)} \sim \xi$.
  $h^{(k)} \sim P_{\mu^{(k)}}^\pi$
  if $\left\|f(h) - f(h^{(k)})\right\| < \varepsilon$ **then**
    $\widehat{M} := \widehat{M} \cup \left\{ \mu^{(k)} \right\}$.
  end if
  if τ **then**
    break
  end if
end for
return $\widehat{M}$

[2] For finite $\mathcal{M}$, the posterior simplifies to $\xi(\mu \mid h, \pi) = \mathbb{P}_\mu^\pi(h)\xi(\mu) / \sum_{\mu' \in \mathcal{M}} \mathbb{P}_{\mu'}^\pi(h)\xi(\mu')$

[3] For episodic problems, we must maintain the sequence of policies used.

sampled set of simulators with a sufficient close statistic is then an approximate sample from our posterior distribution. The first question is what types of statistics we need.

In fact, just as in standard ABC, if the statistic is sufficient, then the samples will be generated according to the posterior.

Corollary 2.1. *If f is a sufficient statistic, then the set $\widehat{M}$ returned by Alg. 1 for $\epsilon = 0$ is a sample from the posterior.*

The (standard) proof is deferred to the appendix. Thus, for $\epsilon = 0$, when the statistic is sufficient, the sampling distribution and the posterior are identical. However, things are not so clear when $\epsilon > 0$.

We now provide a simple theorem which characterises the relation of the approximate posterior to the true posterior, when we use a (not necessarily sufficient) statistic with threshold $\epsilon > 0$. First, we remind the definition of the KL-divergence.

Definition 2.1. *The KL-divergence D between two probability measures ξ, ξ' on $\mathcal{M}$ is*

$$D\left(\xi \parallel \xi'\right) \triangleq \int_{\mathcal{M}} \ln \frac{\mathrm{d}\xi(\mu)}{\mathrm{d}\xi'(\mu)} \, \mathrm{d}\xi(\mu). \tag{2.2}$$

In order to prove meaningful results, we need some additional assumptions on the likelihood function. In this particular case, we simply assume that it is smooth (Lipschitz) with respect to the statistical distance:

Assumption 2.1. *For a given policy π, for any μ, and histories $x, h \in \mathcal{H}$, there exists $L > 0$ such that $\left|\ln \left[\mathbb{P}^\pi_\mu(h) / \mathbb{P}^\pi_\mu(x)\right]\right| \leq L \|f(h) - f(x)\|$.*

We note in passing that this assumption is related to the notion of differential privacy (Dwork & Lei, 2009), from which it was inspired.

We now can state the following theorem, whose proof can be found in the appendix, which generalises the previous corollary.

Theorem 2.1. *Under a policy π and statistic f satisfying Assumption 2.1, the approximate posterior distribution $\xi_\epsilon(\cdot \mid h)$ satisfies:*

$$D\left(\xi(\cdot \mid h) \parallel \xi_\epsilon(\cdot \mid h)\right) \leq (1 + \ln |A^h_\epsilon|) L \epsilon, \tag{2.3}$$

where $A^h_\epsilon \triangleq \{ z \in \mathcal{H} \mid \|f(z) - f(h)\| \leq \epsilon \}$ is the ϵ-ball around the observed history h with respect to the statistical distance and $|A^h_\epsilon|$ denotes its size.

The divergence depends on the statistic in the following ways. Firstly, it approaches 0 as $\epsilon \to 0$. Secondly, it is smaller for smoother likelihoods. However, because of the dependence on the size of the ϵ-ball[4] around the observed statistic, the statistic cannot be arbitrarily smooth. Nevertheless, it may be the case that a sufficient statistic is not required for good performance. Since in reinforcement learning reinforcement learning we are mainly interested in the utility rather than in system identification, we may be able to get good results by using utility-related statistics.

Observation-based statistics A simple idea is to select features on which to calculate statistics. Discounted cumulative feature expectation are especially interesting, due to their connection with value functions (e.g. Puterman, 1994, Sec. 6.9.2). The main drawback is that this adds yet another hyperparameter to tune. In addition, unlike econometrics or bioinformatics, we may not be interested in model identification *per se*, but only in finding a good policy.

Utility-based statistics Quantities related to the utility may be a good match for reinforcement learning. In the simplest case, it may be sufficient to only consider unconditional moments of the utility, which is the approach followed in this paper. However, these may only trivially satisfy Ass. 2.1 for arbitrary policies. Nevertheless, as we shall see, even a very simple such statistic has a reasonably good performance.

2.3. A Hoeffding-based utility statistic

In particular, given a history h including N_{dat} trajectories in the environment, with the i-th trajectory obtaining utility $U^{(i)}$, we obtain a mean estimate $\hat{\mathbb{E}}^{N_{\text{dat}}} U \triangleq \frac{1}{N_{\text{dat}}} U^{(i)}$. We then obtain a history $\hat{h}^{(k)}$ containing N_{trj} trajectories from the sampled environment $\mu^{(k)}$ and construct the mean estimate $\hat{\mathbb{E}}^{N_{\text{trj}}}_k U$. In order to test whether these are close enough, we use the Hoeffding inequality (Hoeffding, 1963). In fact, it is easy to see that, with probability at least $1 - \delta$, $|\,\mathbb{E}^\pi_\mu U - \mathbb{E}^\pi_{\mu^{(k)}} U|$ is lower bounded by:

$$|\hat{\mathbb{E}}^N_{\text{dat}} U - \hat{\mathbb{E}}^{N_{\text{trj}}}_k U| - U_{\max} \sqrt{\frac{\ln(2/\delta)(N_{\text{dat}} + N_{\text{trj}})}{2 N_{\text{dat}} N_{\text{trj}}}}, \tag{2.4}$$

where $U_{\max}$ is the range of the utility function. We then use (2.4) as the statistical distance $\left\|f(h) - f(h^{(k)})\right\|$ between the observed history h and the sampled history $h^{(k)}$. The advantage of using this statistic is that the more data we have, it becomes harder to accept a sample.

[4] For discrete observations this is simply the counting measure of the ball. For more general cases it can be extended to an appropriate measure.

This statistic has two parameters. Firstly, the error probability δ, which does not need to be very small in practice, as the Hoeffding bound is only tight for high-variance distributions. The second parameter is N_{trj}. This does not need to be very large, since it only makes a marginal difference in the bound when $N_{\text{trj}} \gg N_{\text{dat}}$. An illustration of the type of samples obtained with this statistic is given in Figure 1, which shows the dependency of the approximate posterior distribution on the threshold ϵ when conditioned on a fixed amount N_{dat} of training trajectories.

3. ABC reinforcement learning

We now present a simple algorithm for ABC reinforcement learning, based on the ideas explained in the previous section. For any given set of observations and policies, we draw a number of sample environments from the prior distribution. For each environment, we execute the relevant policy and calculate the appropriate statistics. If these are close enough to the observed statistic, the sample is accepted. The next step is to find a good policy for the sampled simulator. As we can draw an arbitrary number of rollouts in the simulator, any type of approximate dynamic programming algorithm can be used. In our experiments, we used LSPI (Lagoudakis & Parr, 2003b), which is simple to program and effective. The hope is that if the approximate posterior sampling is reasonable, then we can take advantage of our prior knowledge of the environment class, to learn a good policy with less data, at the expense of additional computation.

Algorithm 2 ABC-RL

parameters $\mathcal{M}$, ξ, h, π, f
$\tau = \{|\widehat{M}| = 1\}$
$\hat{\mu} = \texttt{ABC-RL-Sample}(\mathcal{M}, \xi, h, \pi, f, \tau)$
return $\hat{\pi} \approx \arg\max_\pi \mathbb{E}^\pi_{\hat{\mu}} U$

A sketch of the algorithm is shown in Alg.2. This has a number of additional parameters that need to be discussed. The most important is the stopping condition τ. The simplest idea, which we use in this paper, is to stop when a single model $\hat{\mu}$ has been generated by `ABC-RL-Sample`.

Then an (approximate) optimal policy for the sampled model $\hat{\mu}$ can be found via an exact (or approximate) dynamic programming algorithm. This simplifies the optimisation step significantly, as otherwise it would be necessary to optimise over multiple models. This particular version of the algorithm can be seen as an ABC variant of Thompson sampling (Strens, 2000; Thompson, 1933).

The exact algorithm to use for the policy optimisation depends largely upon the class of simulators we have. In principle any type of environment can be handled, as long as a simulation-based approximation method can be used to discover a good policy. In *extremis*, direct policy search may be used. However, in the work presented in this paper, we limit ourselves to continuous-state Markov decision processes, for which numerous efficient ADP algorithms exist.

3.1. ABC-LSPI

Let us consider the class of continuous-state, discrete-action Markov decision processes (MDPs). Then, a number of sample-based ADP algorithms can be used to find good policies, such as fitted Q-iteration (FQI) (Ernst et al., 2005) and least-square policy iteration (LSPI) (Lagoudakis & Parr, 2003b), which we use herein.

Since we take an arbitrary number of trajectories from the sampled MDP, an important algorithmic parameter is the number of rollouts N_{rol} to draw. Higher values lead to better approximations, at the expense of additional computation. Finally, since LSPI uses a linear value function[5] approximation, it is necessary to select an appropriate basis for the fit to be good.

The computational complexity of ABC-LSPI depends on the quality of approximation we wish to achieve and on the number of samples required to sample a model with statistics ε-close to those of the data. To reduce computation, if N_{sam} models have been generated without one being accepted, we double ϵ and call `ABC-RL-Sample` again.

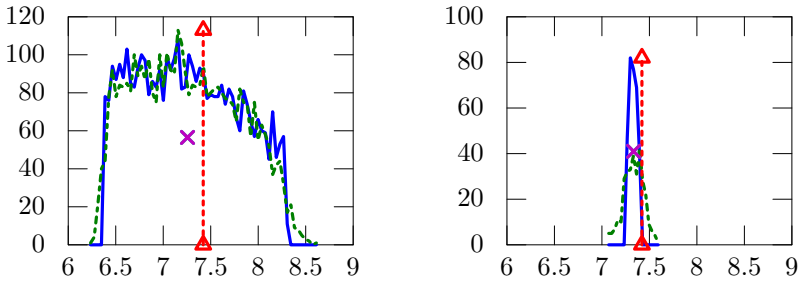
4. Experiments

We performed some experiments to investigate the viability of ABC-RL, with all algorithms implemented using (Dimitrakakis et al., 2007). In these, we compared ABC-LSPI to LSPI. The intuition is that, if ABC can find a good simulator, then we can perform a much better estimation of the value function by drawing a lage number of samples from the simulator, rather than estimating the value function directly from the observations.

4.1. Domains

We consider two domains to illustrate ABC-RL. In both of these domains, we have access to a set of

[5] The value function $V(s)$ is simply the expected utility conditioned on the system state s. We omit details as this is not necessary to understand the framework proposed.



(a) $\varepsilon = 1, N_{\text{dat}} = 10^3$ (b) $\varepsilon = 0.1, N_{\text{dat}} = 10^3$

Figure 1. **Pendulum value distribution**. In both cases, $N_{\text{sam}} = 10^4$ model samples are drawn from the prior and $N_{\text{rol}} = 10^3$ rollouts are performed for each model sample. The vertical dashed line shows the actual value of the policy. The solid and dot-dashed lines show the histograms of real and estimated values of the original policy in the sampled environment. The solid line shows the value estimated using 10^4 rollouts. The dot-dashed line shows the value estimated in the run itself, with N_{trj} rollouts per sample. The $\times$ shows the expected value, averaged over the accepted samples. It can be seen that, while a smaller threshold can result in better accuracy, many fewer samples are accepted.

parametrised simulators $\mathcal{M} = \{ \mu_\theta \mid \theta \in \Theta \}$ for the domains. However, we do not know the true parameters $\theta^* \in \Theta$ of the domains. For ABC, sampled parameters $\theta^{(k)}$ are drawn from a uniform distribution $\mathcal{U}\mathit{nif}(\Theta)$, with $\Theta = \left\{ \theta \in \mathbb{R}^n \;\middle|\; \theta_i \in [\frac{1}{2}\theta_i^*, \frac{3}{2}\theta_i^*] \right\}$.

Mountain car This is a generalised version of the mountain car domain described in Sutton & Barto (1998). The goal is to bring a car to the top of a hill. The problem has 7 parameters: upper and lower bounds on the horizontal position of the car, upper and lower bounds on the car's velocity, upper bounds on the car's forwards and backwards acceleration power, and finally the amount of uniform noise present. The real environment parameters are $\theta^* = (0.5, -1.2, 0.07, -0.07, 0.001, 0.0025, 0.2)$. In this problem, the goal is to reach the right-most horizontal position. The observation consists of the horizontal position and velocity and the reward is -1 at every step until the goal is reached.

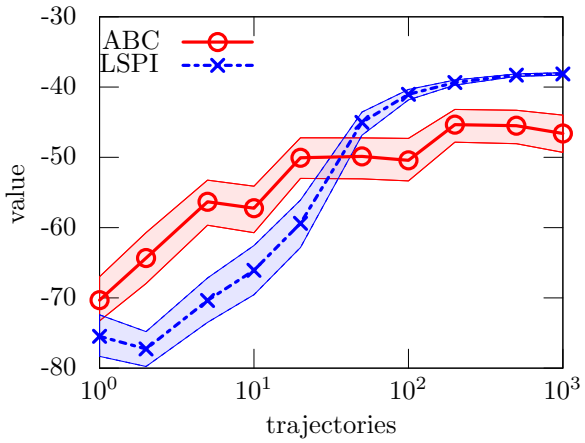
Pendulum This is a generalised version of the pendulum domain (Sutton & Barto, 1998), but without boundaries. The goal of the agent in this environment is to maintain a pendulum upright, using a controller that can switch actions every $0.1s$. The problem has 6 parameters: the pendulum mass, the cart mass, the pendulum length, the gravity, the amount of uniform noise, and the simulation time interval. In this environment, the reward is $+1$ for every step where the pendulum is balanced. The actual environment parameters are $\theta^* = (2.0, 8, 0, 0.5, 9.8, 0.01, 0.01)$.

4.2. Results

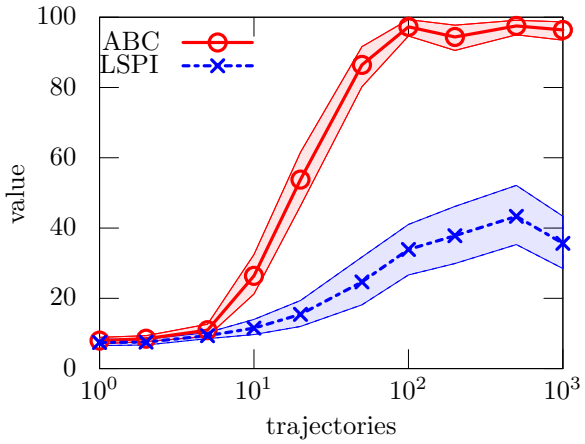
We compared the offline performance of LSPI and ABC-LSPI on the two domains. We first observe N_{dat} trajectories in the real environment drawn using a uniformly random policy. These trajectories are used by both ABC-LSPI and LSPI to estimate a policy. This policy is then evaluated over 10^3 trajectories. The experiment was repeated for 10^2 runs. Since LSPI requires a basis, in both cases we employed a uniform 4×4 grid of RBFs, as well as an additional unit basis for the value function estimation.

The results of the experiment are shown in Fig. 2, where we plot the expected utility (with a discount factor $\gamma = 0.99$) of the policy found as the number of trajectories increase. Both LSPI and ABC-LSPI manage to find an improved policy with more data. However, the source of their improvement is different. In the case of LSPI, the additional data leads to better estimation of the value function. In ABC-LSPI, the additional data leads to a better sampled model. The value function is then estimated using a large number of rollouts in the sampled model. The CPU time taken by ABC ranges in 20 to 40s, versus 0.05 to 30s for pure LSPI, depending on the amount of training data. This is due to the additional overhead of sampling as well as the increased amount of rollouts used for ADP.

In general, the ABC approach quickly reaches a good performance, but then has little improvement. This effect is particularly prominent in the Mountain Car domain (Fig. 2(a)), where it is significantly worse asymptotically than LSPI. This can be attributed to the fact that even though more data is available, the number of samples drawn from the prior is not sufficient for a



(a) Mountain Car



(b) Pendulum

Figure 2. **Off-line performance.** For $N_{\text{sam}} = 10^3$, $\varepsilon = 10^{-2}$, $N_{\text{trj}} = 10^2$, $N_{\text{rol}} = 2 \cdot 10^3$, $\gamma = 0.99$. The data are averaged over 10^2 runs, with each run being evaluated with 10^3 trajectories. The shaded regions show 95% bootstrap confidence intervals from 10^3 bootstrap samples.

good model to be found. In fact, upon investigation we noticed that although most model parameters were reliably estimated, there was a difficulty in estimating the goal location from the given trajectories. This was probably the main reason why ABC didn't reach optimal performance in this case. However, it may be possible to improve upon this result with a more efficient sampling scheme, or a statistic that is closer to sufficiency than the simple utility-based statistic we used.

On the other hand, the performance is significantly better than LSPI in the pendulum environment (Fig. 2(b)). There are two possible reason for this. Firstly, ABC-LSPI not only uses more samples for the value function estimation, but also better distributed samples, as it estimates the value function by drawing trajectories starting from uniformly drawn states in the sampled environment. Secondly, and perhaps more importantly, that even for very differently parametrised pendulum problems the optimal policies on the pendulum domain are quite similar. Thus, even if ABC only samples a very approximate simulator, its optimal policy is going to be close to that of the real environment.

5. Conclusion

We presented an extension of ABC, a likelihood-free method for approximate Bayesian computation, to controlled dynamical systems. This method is particularly interesting for domains where it is difficult to specify an appropriate probabilistic model, and where computation is significantly cheaper than data collection. It is in principle generally applicable to any type of reinforcement learning problem, including continuous, partially observable and multi-agent domains. We also introduce a general theorem for the quality of the approximate ABC posterior distribution, which can be used for further analysis of ABC methods.

We then applied ABC inference to reinforcement learning. This involves using simulation both to estimate approximate posterior distributions and to find good policies. Thus, ABC-RL can be simultaneously seen as an extension of ABC inference to control problems and an extension of approximate dynamic programming methods to likelihood-free approximate Bayesian inference. The main advantage is when have no reasonable probabilistic model, but we do have access to a parametrised set of simulators, which contain good approximations to the real environment. This is frequently the case in complex control problems. However, we see that ABC-RL (specifically ABC-LSPI) is competitive with pure LSPI even in problems with low dimensionality where LSPI is expected to perform quite well.

ABC-RL appears a viable approach, even with a very simple sampling scheme, and a utility-based statistic. In future work, we would like to investigate more elaborate ABC schemes such as Markov chain Monte Carlo, as well as statistics that are closer to sufficient, such as discounted feature expectations and conditional utilities. This would enable us to examine its performance in more complex problems where the practical advantages of ABC would be more evident. However, we believe that the results are extremely encouraging and that the ABC methodology has great potential in the field of reinforcement learning.

A. Collected proofs

Proof of Remark 2.1. Let $h = (x^{T+1}, a^T, r^T)$. Using induction,

$$\mathbb{P}_\mu^\pi(h) = \prod_{t=0}^{T} \mu_t(x_{t+1})\pi_t(a_t).$$

Replacing in the posterior calculation (A.1) we obtain:

$$\xi(B \mid h, \pi) = \frac{\int_B \prod_{t=0}^T \mu_t(x_{t+1}) \,\mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \prod_{t=0}^T \mu_t(x_{t+1}) \,\mathrm{d}\xi(\mu)} \tag{A.1}$$

since the $\prod_{t=0}^T \pi_t(a_t)$ terms can be taken out of the integrals and cancel out. □

Proof of Corollary 2.1. By definition, a sufficient statistic $f : \mathcal{H} \to \mathcal{W}$ has the following property:

$$\forall \mu, \pi : \quad \mathbb{P}_\mu^\pi(h) = \mathbb{P}_\mu^\pi(h') \qquad \text{iff } f(h) = f(h'). \tag{A.2}$$

The probability of drawing a model in $B \subset \mathcal{M}$ is:

$$\begin{aligned}&\frac{\int_B \sum_{z\in\mathcal{H}} \mathbb{I}\left\{f(z) = f(h)\right\} \mathbb{P}_\mu^\pi(z) \,\mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \sum_{z\in\mathcal{H}} \mathbb{I}\left\{f(z) = f(h)\right\} \mathbb{P}_\mu^\pi(z) \,\mathrm{d}\xi(\mu)}\\ =&\frac{\int_B \mathbb{P}_\mu^\pi(h) \,\mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \mathbb{P}_\mu^\pi(h) \,\mathrm{d}\xi(\mu)} = \xi(B \mid h, \pi),\end{aligned} \tag{A.3}$$

due to (A.2). □

Proof of Theorem 2.1. For notational simplicity, we introduce $\phi(\cdot) = \int_{\mathcal{M}} \mathbb{P}_\mu^\pi(\cdot) \,\mathrm{d}\xi(\mu)$ for the marginal prior measure on $\mathcal{H}$, also omitting the dependency on π. Then the ABC posterior $\xi_\epsilon(B \mid h)$ equals:

$$\begin{aligned}&\frac{\int_B \sum_{z\in\mathcal{H}} \mathbb{I}\left\{\|f(z) - f(h)\| < \epsilon\right\} \mathbb{P}_\mu^\pi(z) \,\mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \sum_{z\in\mathcal{H}} \mathbb{I}\left\{\|f(z) - f(h)\| < \epsilon\right\} \mathbb{P}_\mu^\pi(z) \,\mathrm{d}\xi(\mu)}\\ =&\frac{\int_B \mathbb{P}_\mu^\pi(A_\epsilon^h) \,\mathrm{d}\xi(\mu)}{\int_{\mathcal{M}} \mathbb{P}_\mu^\pi(A_\epsilon^h) \,\mathrm{d}\xi(\mu)} = \frac{\int_B \mathbb{P}_\mu^\pi(A_\epsilon^h) \,\mathrm{d}\xi(\mu)}{\phi(A_\epsilon^h)}.\end{aligned} \tag{A.4}$$

From Definition 2.1:

$$\begin{aligned}&D\left(\xi(\cdot \mid h) \parallel \xi_\epsilon(\cdot \mid h)\right) = \int_{\mathcal{M}} \ln \frac{\mathrm{d}\xi(\mu \mid h)}{\mathrm{d}\xi_\epsilon(\mu \mid h)} \,\mathrm{d}\xi(\mu)\\ &\overset{(a)}{=} \int_{\mathcal{M}} \ln \left(\frac{\mathbb{P}_\mu^\pi(h)}{\mathbb{P}_\mu^\pi(A_\epsilon^h)} \times \frac{\phi(A_\epsilon^h)}{\phi(h)}\right) \mathrm{d}\xi(\mu)\\ &= \int_{\mathcal{M}} \ln \frac{\mathbb{P}_\mu^\pi(h)}{\mathbb{P}_\mu^\pi(A_\epsilon^h)} \,\mathrm{d}\xi(\mu) + \int_{\mathcal{M}} \ln \frac{\phi(A_\epsilon^h)}{\phi(h)} \,\mathrm{d}\xi(\mu)\\ &\overset{(b)}{\leq} \int_{\mathcal{M}} \ln \frac{\mathbb{P}_\mu^\pi(h)}{\min_{z\in A_\epsilon^h} \mathbb{P}_\mu^\pi(z)} \,\mathrm{d}\xi(\mu) + \int_{\mathcal{M}} \ln \frac{\phi(A_\epsilon^h)}{\phi(h)} \,\mathrm{d}\xi(\mu)\\ &\overset{(c)}{\leq} \int_{\mathcal{M}} \left|\ln \frac{\mathbb{P}_\mu^\pi(h)}{\min_{z\in A_\epsilon^h} \mathbb{P}_\mu^\pi(z)}\right| \mathrm{d}\xi(\mu) + \int_{\mathcal{M}} \left|\ln \frac{\phi(A_\epsilon^h)}{\phi(h)}\right| \mathrm{d}\xi(\mu)\\ &\overset{(d)}{\leq} L\epsilon + \left|\ln \frac{\phi(A_\epsilon^h)}{\phi(h)}\right| \overset{(e)}{\leq} L\epsilon(1 + \ln |A_\epsilon^h|).\end{aligned}$$

Equality (a) follows from equations (A.3) and (A.4). Inequality (b) follows from the fact that $\mathbb{P}_\mu^\pi(A_\epsilon^h) = \sum_{z\in A_\epsilon^h} \mathbb{P}_\mu^\pi(z) \geq \min_{z\in A_\epsilon^h} \mathbb{P}_\mu^\pi(z)$, while (c) follows from $|x| \geq x$. For (d), first note that for any $z \in A_\epsilon^h$, by the definition of A_ϵ^h, $|\ln[\mathbb{P}_\mu^\pi(h)/\,\mathbb{P}_\mu^\pi(z)]| \leq L\epsilon$, by Assumption 2.1. Thus the first integral is bounded by $\int_{\mathcal{M}} \xi(\mu) = \xi(\mathcal{M}) = 1$. Similarly, the $|\cdot|$ term in the second integral is independent of μ and so is taken out. For (e), the same assumption gives that $\phi(z) = \int_{\mathcal{M}} \mathbb{P}_{\mu,\pi}(z) \,\mathrm{d}\xi(\mu) \leq \exp(L\epsilon)\phi(h)$ for any $z \in A_\epsilon^h$ so, $\ln[\phi(A_\epsilon^h)/\phi(h)] \leq L\epsilon \ln |A_\epsilon^h|$. Finally, as $h \in A_\epsilon^h$, $\phi(A_\epsilon^h) \geq \phi(h)$ and we obtain the final result. □

References

Agrawal, Shipra and Goyal, Navi. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT 2012*, 2012.

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

Bertsekas, Dimitri P. *Nonlinear Programming*. Athena Scientific, 1999.

Bertsekas, Dimitri P. Rollout algorithms for constrained dynamic programming. Technical Report LIDS 2646, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass., 2006.

Bertsekas, Dimitri P. and Tsitsiklis, John N. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Castro, Pablo Samuel and Precup, Doina. Using linear programming for Bayesian exploration in Markov decision processes. In Veloso, Manuela M. (ed.), *IJCAI*, pp. 2437–2442, 2007.

Csilléry, K., Blum, M.G.B., Gaggiotti, O.E., François, O., et al. Approximate Bayesian computation (ABC) in practice. *Trends in ecology & evolution*, 25(7):410–418, 2010.

Dean, Thomas A and Singh, Sumeetpal S. Asymptotic behaviour of approximate bayesian estimators. *arXiv preprint arXiv:1105.3655*, 2011.

DeGroot, Morris H. *Optimal Statistical Decisions*. John Wiley & Sons, 1970.

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

Dimitrakakis, Christos and Lagoudakis, Michail G. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, September 2008. doi: 10.1007/s10994-008-5069-3. Presented at ECML'08.

Dimitrakakis, Christos, Tziortziotis, Nikolaos, and Tossou, Aristide. Beliefbox: A framework for statistical methods in sequential decision making. `http://code.google.com/p/beliefbox/`, 2007.

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

Dwork, Cynthia and Lei, Jing. Differential privacy and robust statistics. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pp. 371–380. ACM, 2009.

Ernst, Damien, Geurts, Pierre, and Wehenkel, Louis. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Gabillon, Victor, Lazaric, Alessandro, Ghavamzadeh, Mohammad, and Scherrer, Bruno. Classification-based policy iteration with a critic. In *ICML 2011*, 2011.

Geweke, J. Using simulation methods for Bayesian econometric models: inference, development, and communication. *Econometric Reviews*, 18(1):1–73, 1999.

Hoeffding, Wassily. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

Jasra, Ajay, Singh, Sumeetpal S, Martin, James S, and McCoy, Emma. Filtering via approximate bayesian computation. *Stat. Comput*, 2010.

Kaufmanna, Emilie, Korda, Nathaniel, and Munos, Rémi. Thompson sampling: An optimal finite time analysis. In *ALT-2012*, 2012.

Kolter, J. Zico and Ng, Andrew Y. Near-Bayesian exploration in polynomial time. In *ICML 2009*, 2009.

Lagoudakis, M. and Parr, R. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML*, pp. 424, 2003a.

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

Marin, J.M., Pudlo, P., Robert, C.P., and Ryder, R.J. Approximate Bayesian computational methods. *Statistics and Computing*, pp. 1–14, 2011.

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

Poupart, Pascal and Vlassis, Nikos. Model-based Bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

Puterman, Marting L. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New Jersey, US, 1994.

Ross, Stephane, Chaib-draa, Brahim, and Pineau, Joelle. Bayes-adaptive POMDPs. In Platt, J.C., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20*, Cambridge, MA, 2008. MIT Press.

Savage, Leonard J. *The Foundations of Statistics*. Dover Publications, 1972.

Strens, Malcolm. A Bayesian framework for reinforcement learning. In *ICML 2000*, pp. 943–950, 2000.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

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

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, M.P.H. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

Vlassis, N., Ghavamzadeh, M., Mannor, S., and Poupart, P. *Reinforcement Learning*, chapter Bayesian Reinforcement Learning, pp. 359–386. Springer, 2012.

Wu, Feng, Zilberstein, Shlomo, and Chen, Xiaoping. Rollout sampling policy iteration for decentralized POMDPs. In *The 26th conference on Uncertainty in Artificial Intelligence (UAI 2010)*, Catalina Island, CA, USA, July 2010.