Smarter Sampling in Model-Based Bayesian Reinforcement Learning

Pablo Samuel Castro and Doina Precup

School of Computer Science
McGill University
{pcastr,dprecup}@cs.mcgill.ca

Abstract. Bayesian reinforcement learning (RL) is aimed at making more efficient use of data samples, but typically uses significantly more computation. For discrete Markov Decision Processes, a typical approach to Bayesian RL is to sample a set of models from an underlying distribution, and compute value functions for each, e.g. using dynamic programming. This makes the computation cost per sampled model very high. Furthermore, the number of model samples to take at each step has mainly been chosen in an ad-hoc fashion. We propose a principled method for determining the number of models to sample, based on the parameters of the posterior distribution over models. Our sampling method is local, in that we may choose a different number of samples for each state-action pair. We establish bounds on the error in the value function between a random model sample and the mean model from the posterior distribution. We compare our algorithm against state-of-the-art methods and demonstrate that our method provides a better trade-off between performance and running time.

1 Introduction

Reinforcement learning (RL) attempts to find an optimal way of behaving in an unknown environment, often assumed to be a Markov Decision Process (MDP). By interacting with the environment, an RL agent learns more about its dynamics and rewards; this information can be used to shape the agent's behavior or policy. Traditional RL methods are typically based on estimating the expected value of the agent's long-term return (also called a value function). However, in some applications one would also like to have an estimate of the agent's uncertainty, in addition to the expectation of the returns. Bayesian RL methods attempt to provide such estimates by maintaining either a distribution over models, e.g.(Asmuth et al., 2009; Dearden et al., 1999; Poupart et al., 2006; Price & Boutilier, 2003; Strens, 2000), or a distribution over value functions, e.g. (Engel et al., 2003). The first category of methods is often called model-based Bayesian RL and will be the focus of our paper. The Bayesian approach allows incorporating prior knowledge about the MDP (if available), in the form of a prior. Bayesian approaches have also been proposed as a method that can potentially use small amounts of data; hence, they could be useful in applications where the data is difficult to obtain. Several Bayesian RL method emphasize the use of the distribution as a tool for formulating good exploration policies, e.g. (Duff, 2003; Asmuth et al., 2009; Wang et al., 2005). However, recent work (Kolter & Ng, 2009) has cast doubt on the effectiveness of Bayesian exploration. However, even if exploration is not the main emphasis, having uncertainty estimates in the performance of a policy is still quite useful.

Most model-based Bayesian RL methods work by maintaining a distribution over model parameters, which is initialized with a prior and updated based on data obtained from the real environment. The algorithms usually sample a batch of models from this distribution and the value function is computed for each of these sampled models. This can make the computation cost for each iteration very high, especially for large systems. The number of sampled models is often chosen in an ad-hoc fashion, based on computation time constraints. Models are re-sampled periodically, either at fixed intervals, or when their likelihood drops below a certain threshold. Recent work has attempted to make this choice in a more principled way. In (Asmuth et al., 2009), the authors propose an algorithm (BOSS) which aims to ensure enough exploration; they prove that if one samples $\Theta(\frac{1}{\delta}\ln\frac{1}{\delta})$ models, then with probability at least $1-\delta$ one of these models will yield a value function that is optimistic compared to the true underlying model. They also provide theoretically a value for the number of data samples that have to be gathered before new models should be sampled. However, the computation of both the number of models and the number of samples is difficult and for their empirical results, the authors resort to choosing these values manually.

In this paper, we propose a new method for determining the number of model samples to take, in the context of discrete MDPs, based on the parameters of the posterior distribution over models. Furthermore, we argue that one should sample locally: the number of transition models to sample for each state-action pair should depend on the statistical properties of the posterior distribution for that particular state-action pair, rather than being fixed over the entire MDP. We overcome the difficulty of combining these different numbers of transition samples by constructing a *merged* MDP as in (Asmuth et al., 2009). We use an optimistic approach to resolve the exploration-exploitation dilemma. We provide a method for dynamically determining when to re-sample and re-compute the solution to the merged MDP. This decision is based on the difference between the mean of the posterior last used for sampling and the mean of the current posterior distribution. Our method exploits the statistical properties of the posterior distribution and the proposed score can be computed very quickly.

We motivate our approach by first providing theoretical bounds for the difference in value function between a random model sampled from a Dirichlet distribution and the mean of the distribution. The tightness of the bounds depends directly on the variance of the posterior distribution; hence, the variance directly affects the decision of when and how much to sample in our proposed algorithms.

The paper is structured as follows. Sec. 2 presents necessary background and notation. In Sec. 3 we present bounds on the difference between the value function of the mean of a distribution over MDPs, compared to a randomly sampled model. Sec. 4 presents two algorithms for determining the number of models and the frequency of sampling them, based on these bounds. Sec. 5 contains an empirical comparison of the proposed algorithms against other Bayesian RL methods. Sec. 6 contains a discussion and avenues for future work.

2 Background

A finite Markov Decision Process (MDP) $M = \langle S, A, P, R \rangle$ consists of a finite set of states S; a finite set of actions A; a transition function $P : S \times A \longrightarrow Dist(S)$, where $Dist(X)$ is the set of distributions over X; and a reward function $R : S \times A \times S \longrightarrow \mathbb{R}$ (see (Puterman, 1994) for more details). A *policy* is a function $\pi : S \rightarrow Dist(A)$. The value of a state $s \in S$ given a policy π is defined as $V^{\pi}(s) = \mathbb{E}^{\pi}\left(\sum_{i=0}^{\infty} \gamma^i r_i\right)$, where $0 \leq \gamma < 1$ is a discount factor and r_i a random variable representing the reward received at time step i when starting from state s and choosing actions according to policy π. The values for all states can be computed by solving the following system of Bellman linear equations:

$$V^{\pi}(s) = \sum_{s' \in S} P(s, \pi(s))(s') \left[R(s, \pi(s), s') + \gamma V^{\pi}(s')\right]$$

The optimal value function is defined as: $V^*(s) = \max_{\pi} V^{\pi}(s), \forall s \in S$ and obeys the following system of non-linear equations:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P(s,a)(s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

If the model of the MDP (i.e. R and P) is known, several well-known methods can be used to solve this system, e.g. dynamic programming and linear programming.

In reinforcement learning (RL) (Sutton & Barto, 1998) the model of the MDP is usually assumed to be unknown. Traditional model-based RL methods rely on maintaining a model estimate which is updated based on the data obtained by the agent in its interaction with the environment. In Bayesian RL, instead of maintaining just one model, one maintains a distribution over models. In this paper we assume that the reward model is known; this is often the case in RL, as rewards are provided by the designer of the system to define the task at hand. The transition model P, on the other hand, is unknown.

For any $s, s' \in S$ and $a \in A$ let $\theta(s,a,s')$ be a random variable representing the probability of a particular model for the transition probability from s to s' under a. Given a prior distribution $Pr(\theta)$ (representing initial knowledge about θ) and an observed transition $s \rightarrow^a s'$, one can compute the posterior distribution $Pr(\theta | s, a, s')$ using Bayes' Theorem. Usually, the prior and posterior distributions are chosen to be in the same family of distributions. For discrete MDPs, the preferred choice is to use a Dirichlet distribution as a prior/posterior, which is a parameterized distribution conjugate to the multinomial distribution. This is a natural choice since the parameters of a Dirichlet distribution function as "counts" (i.e. the parameter for $\theta(s,a,s')$ can be interpreted as the number of times the transition $s \rightarrow^a s'$ has been observed). A typical approach is to sample at each step a number of models from the distribution over models, compute the optimal value function for each sample model, and then use these values to choose the next action.

The Dirichlet distribution is parameterized by a set of real numbers $\alpha = (\alpha_1, \cdots, \alpha_N)$, where N is the order of the Dirichlet distribution. Let $\alpha_0 = \sum_{i=1}^{N} \alpha_i$. The Dirichlet is the conjugate prior of the Multinomial distribution (a multivariate generalization of the binomial distribution) with parameters $p_1 = \alpha_1/\alpha_0$, $\cdots$, $p_N = \alpha_N/\alpha_0$. In a finite MDP, a Dirichlet distribution $Dir(\alpha^{(s,a)})$ of order $N = |S|$ can be maintained for each state-action pair (s,a), where $\alpha_{s'}^{(s,a)}$ is the parameter for transition $s \rightarrow^{a} s'$. We refer to the *mean (or mean model)* of a Dirichlet distribution $Dir(\alpha^{(s,a)})$ as the multinomial distribution $\mu_{(s,a)} \in Dist(S)$ where $\mu_{(s,a)(s_i)} = \alpha_i^{(s,a)} / \alpha_0^{(s,a)}$. The marginals of this multinomial have variance given by $\sigma^2_{(s,a)(s_i)} = \frac{\alpha_{(s,a)(s_i)}(\alpha_0^{(s,a)} - \alpha_{(s,a)(s_i)})}{\left(\alpha_0^{(s,a)}\right)^2 (\alpha_0^{(s,a)}+1)}$. Throughout the learning process we will maintain a set of Dirichlet distributions parameterized by state-action pairs: $\{Dir(\alpha_{(s,a)})\}_{s \in S. a \in A}$.

3 Bounding the Value Function

As the agent gathers experience by interacting with the environment, the expected distance between a random sample and the mean model should go down. As this expected distance goes down, the number of required samples should also go down. Furthermore, once enough experience has been obtained, the change in model estimates should change less and less, implying that the need to resample should decrease. These are the main ideas of the algorithms presented in the next section, but we begin by providing a probabilistic bound on the difference between a random sample and the mean model, formalizing the intuition just mentioned.

Let $\alpha^{(s,a)}$ be the parameters for $s \in S$ and $a \in A$ with $\alpha_0^{(s,a)} = \sum_{s' \in S} \alpha_{s'}^{(s,a)}$; $\mu_{(s,a)(s')}$ and $\sigma^2_{(s,a)(s')}$ are the expected value and variance, respectively, for any $s' \in S$.

A well-known way to measure the difference between two probability distributions over the same domain is the total variation distance. Given two state distributions $P, Q \in Dist(S)$, their total variation distance (TV) is defined as:

$$TV(P,Q) = \frac{1}{2} \sum_{s \in S} |P(s) - Q(s)|$$

Clearly $0 \leq TV(P,Q) \leq 1$ for any P,Q. We can now easily obtain the following two inequalities (stated without proof), where V_1 and V_2 are the value functions for two MDPs that differ only in their transition probabilities (P_1 and P_2, respectively):

Lemma 1. *For any policy* π*,*

$$|V_1^{\pi}(s) - V_2^{\pi}(s)| \leq \frac{2\gamma R_{max}}{1-\gamma} \max_{s \in S} \max_{a \in A} TV(P_1(s,a), P_2(s,a))$$

For the optimal value function:

$$|V_1^{*}(s) - V_2^{*}(s)| \leq \frac{2\gamma R_{max}}{1-\gamma} \max_{s \in S} \max_{a \in A} \max_{b \in A} TV(P_1(s,a), P_2(s,b))$$

Both these inequalities can be bounded by the trivial upper bound $\frac{2\gamma R_{max}}{1-\gamma}$, which is very loose. We would like to be able to tighten the bounds of Lemma 1 by using information from the Dirichlet distribution. In particular, given a sampled transition model $X_{(s,a)}$

from $Dir(\alpha^{(s,a)})$ and a real number $m > 1$, we would like to know with what probability $2TV(\mu_{(s,a)}, X_{(s,a)}) < 1/m$. One approach is to examine, for each $s' \in S$, with what probability $|X_{(s,a)(s')} - \mu_{(s,a)(s')}| < 1/(Nm)$, where $N = |S|$.

A simple application of Chebyshev's inequality yields the following result.

$$Pr\left(|X_{(s,a)(s')} - \mu_{(s,a)(s')}| \geq \frac{1}{mN}\right) \leq \sigma^2_{(s,a)(s')}(mN)^2$$

Now we can obtain a lower bound for the total variation being less than $1/2m$ as follows:

$$\begin{aligned} Pr\left(2TV(X_{(s,a)}, \mu_{(s,a)}) < \frac{1}{m}\right) &\geq \prod_{s' \in S} Pr\left(|X_{(s,a)(s')} - \mu_{(s,a)(s')}| < \frac{1}{mN}\right) \\ &\geq \prod_{s' \in S}\left(1 - \sigma^2_{(s,a)(s')}(mN)^2\right) \geq 1 - \sum_{s' \in S}\left(\sigma^2_{(s,a)(s')}(mN)^2\right) \\ &= 1 - (mN)^2 \sum_{s' \in S} \sigma^2_{(s,a)(s')} \end{aligned} \quad (1)$$

where the second to last line follows by Weierstrass' product inequality. Let us examine the sum of the variances:

$$\begin{aligned} \sum_{s' \in S} \sigma^2_{(s,a)(s')} &= \sum_{s' \in S} \frac{\alpha^{(s,a)}_{s'}\left(\alpha^{(s,a)}_0 - \alpha^{(s,a)}_{s'}\right)}{\left(\alpha^{(s,a)}_0\right)^2\left(\alpha^{(s,a)}_0 + 1\right)} = \frac{\sum_{s' \in S}\alpha^{(s,a)}_{s'}\left(\alpha^{(s,a)}_0 - \alpha^{(s,a)}_{s'}\right)}{\left(\alpha^{(s,a)}_0\right)^2\left(\alpha^{(s,a)}_0 + 1\right)} \\ &= \frac{\alpha^{(s,a)}_0 \sum_{s' \in S}\alpha^{(s,a)}_{s'}}{\left(\alpha^{(s,a)}_0\right)^2\left(\alpha^{(s,a)}_0 + 1\right)} - \frac{\sum_{s' \in S}\left(\frac{\alpha^{(s,a)}_{s'}}{\alpha^{(s,a)}_0}\right)^2}{\alpha^{(s,a)}_0 + 1} = \frac{1}{\alpha^{(s,a)}_0 + 1}\left(1 - \sum_{s' \in S}\mu^2_{(s,a)(s')}\right) \end{aligned}$$

Therefore we have:

$$Pr\left(2TV(X_{(s,a)}, \mu_{(s,a)}) < \frac{1}{m}\right) \geq 1 - \frac{(mN)^2}{\alpha^{(s,a)}_0 + 1}\left(1 - \sum_{i=1}^{N}\mu^2_{(s,a)(s')}\right) \quad (2)$$

Note that if $\alpha^{(s,a)}_0 + 1 \gg (mN)^2$ we get better bounds on the total variation distance. In other words, in order to have tighter bounds on the Total Variation distance between a sampled model and the mean model, we need to have chosen action a from state s enough times in the learning process.

4 Algorithms for Smart Sampling

In this section we use the result above to construct two algorithms that determine the number of sampled models, as well as to decide when to sample new models.

The previous section indicates that the number of sampled models should depend directly on the variance of the underlying model distribution. Given a multinomial sample $X = (X_1, \cdots, X_N) \sim Dir(\alpha^{(s,a)})$, the marginals $X_{s'}$ are Binomial distributions with expected value $\mu_{(s,a)(s')}$ and variance $\sigma^2_{(s,a)(s')}$. Given a set of K model samples $\{X^i\}_{i=1}^K$, for each of the marginals the mean model can be computed as follows:

$$\hat{\mu}^K_{(s,a)(s')} = \frac{1}{K}\sum_{i=1}^{K} X^i_{s'}$$

Additionally, for each of the marginals the variance can be computed as follows:

$$\begin{aligned}\left(\hat{\sigma}^K_{(s,a)(s')}\right)^2 &= Var\left(\frac{1}{K}\sum_{i=1}^{K} X^i_{s'}\right) = \frac{1}{K^2} Var\left(\sum_{i=1}^{K} X^i_{s'}\right)\\ &= \frac{1}{K^2}\sum_{i=1}^{K}\left(\sigma^2_{(s,a)(s')}\right) \qquad \text{Since the } X^i\text{s are drawn independently}\\ &= \frac{\sigma^2_{(s,a)(s')}}{K} \qquad \text{Since the } X^i\text{s are identically distributed}\end{aligned} \tag{3}$$

By the strong law of large numbers, $\lim_{K\to\infty}\hat{\mu}^K_{(s,a)(s')} = \mu_{(s,a)(s')}$, but we would like to determine how many model samples would be "good enough". To achieve this we try to bring the variance for each of the marginals below some constant ε: $\left(\hat{\sigma}^K_{(s,a)(s')}\right)^2 \leq \varepsilon$. To obtain this, for each state-action pair (s,a) we must then set the number of sampled models as follows:

$$K_{(s,a)} = \max_{s'\in S}\left\lceil \frac{\sigma^2_{(s,a)(s')}}{\varepsilon}\right\rceil$$

Although the mean model is readily obtainable from the known posterior parameters, a lower variance is a good indication that we have a good enough approximation of the posterior distribution.

Based on this bound, each state-action pair may have a different number of required samples. In previous methods, if K was the desired number of samples, then K model samples were taken from each state-action pair, and these were combined in the obvious way to form K sampled models. Our situation is slightly more complicated and we cannot form a set of distinct models. We follow the approach of (Asmuth et al., 2009) and construct a merged MDP $m^\#$. The state space in $m^\#$ is the same as the original MDP, but we expand the actions. For each state-action pair (s,a), let $\{T_{(s,a)}(i)\}_{1\leq i\leq K_{(s,a)}}$ be the set of sampled models from (s,a). Each state s in MDP $m^\#$ has $\sum_{a\in A} K_{(s,a)}$ actions. Action $a_{i,j}$, for $1 \leq i \leq K_{(s,j)}$ and $1 \leq j \leq A$, corresponds to the sampled transition $T_{(s,j)}(i)$. As in (Asmuth et al., 2009), we assume the rewards are known in advance for simplicity and for sake of comparison, but the uncertainty about them is encoded in the transitions.

In (Asmuth et al., 2009), if in $m^\#$ the optimal action choice from state s is $a_{i,j}$, then the agent chooses action a_j in the real MDP. This is an optimistic approach that is the basis of their exploration strategy. With this exploration policy the authors can determine a formal choice of K to obtain a near-optimal behavior with high probability. We follow this optimistic exploratory strategy in our approach.

As in most previous work, we do not sample models at every iteration. Instead, we fix a number B beforehand and resample every time a state-action pair has been tried B times during learning. In (Asmuth et al., 2009) the authors determine what the value of B needs to be in order for the model distribution to be "close" to the true model. Although proven theoretically, in practice they set this value manually. We follow this approach for our initial algorithm: SmartSampler.

Algorithm 1. SmartSampler(ε, B)

```
1: Choose a starting state $s_1$
2: For all $(s,a) \in S \times A$, $qCounts(s,a) \leftarrow 0$
3: $reSample \leftarrow TRUE$
4: for all timesteps $t = 1,2,3,\cdots$ do
5:    if $reSample$ then
6:       For all $(s,a) \in S \times A$, sample $K_{(s,a)}$ transitions from the posterior
7:       Combine all the samples into the merged MDP $m^{\#}$
8:       Solve $m^{\#}$ and extract $\pi^{\#}$
9:       $reSample \leftarrow FALSE$
10:   end if
11:   Extract $a_t$ from $\pi^{\#}(s_t)$
12:   Perform action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
13:   $qCounts(s_t,a_t) \leftarrow qCounts(s_t,a_t) + 1$
14:   Update posterior based on $(s_t,a_t,r_t,s_{t+1})$
15:   if $qCounts(s_t,a_t) = B$ then
16:      $reSample \leftarrow TRUE$
17:   end if
18: end for
```

The SmartSampler algorithm chooses how many model samples to take for each state-action pair, based on the parameters of the posterior distribution. Establishing *when* we should resample and re-compute a solution to the resulting MDP $m^{\#}$ is still unresolved. We propose determining when to sample by examining the change in the distribution of the mean models of the posteriors. We use the standard score as inspiration for determining this change. Specifically, we wish to determine how many standard deviations the mean model of the current posterior is from the mean model of the posterior used the last time model sampling was performed. For any $(s,a) \in S \times A$, let $P_t(s,a)$ be the mean transition distribution for (s,a) of the posterior at time t. Let timestep t be the last time we sampled a model from our posterior to obtain $m^{\#}$. Given a maximum threshold parameter δ, we will resample at time step t' only when

$$\sum_{s' \in S} \frac{|P_t(s,a)(s') - P_{t'}(s,a)(s')|}{\sigma_{(s,a)(s')}} > \delta$$

Note that the standard deviation $\sigma_{(s,a)(s')}$ used is the one computed from the posterior at timestep t (*i.e.* when we last sampled models). This measure of change takes into account both changes in the transition distribution as well as changes in the variance. In this way, if we start with a confident prior that is already quite close to the true

Algorithm 2. SmarterSampler(ε, δ)

```
1: Choose a starting state $s_1$
2: $reSample \leftarrow TRUE$
3: $lastSamp \leftarrow 1$
4: for all timesteps $t = 1, 2, 3, \cdots$ do
5:     if $reSample$ then
6:         For all $(s,a) \in S \times A$, sample $K_{(s,a)}$ transitions from $\alpha^{(s,a)}$
7:         Combine all the samples into the merged MDP $m^{\#}$
8:         Solve $m^{\#}$ and extract $\pi^{\#}$
9:         $lastSamp \leftarrow t$
10:    end if
11:    Extract $a_t$ from $\pi^{\#}(s_t)$
12:    Perform action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
13:    Update posterior based on $(s_t, a_t, r_t, s_{t+1})$
14:    if $\sum_{s' \in S} \frac{|P_t(s_t,a_t)(s') - P_{lastSamp}(s_t,a_t)(s')|}{\sigma_{(s,a)(s')}} > \delta$ then
15:        $reSample \leftarrow TRUE$
16:    end if
17: end for
```

model, we will resample very infrequently. Furthermore, as the algorithm gathers more information, the number of times we need to resample decreases. Algorithm Smarter-Sampler summarizes this approach.

5 Experimental Results

We compare our algorithm only against the BOSS algorithm of (Asmuth et al., 2009), because it is similar in spirit and the efficacy of BOSS against other algorithms was already established in (Asmuth et al., 2009). Additionally, we compare against a Naive Bayesian algorithm that uses the same parameters as BOSS. For this approach, K indicates the number of global model samples to take (as in BOSS); however, the sampled models are not combined into a merged MDP $m^{\#}$; instead, each one is solved independently and the maximum value for each state-action pair is used. The B parameter indicates how many iterations should pass before resampling; thus, every B iterations the naive algorithm will sample K models. All algorithms were implemented in Matlab; for fairness and consistency, the algorithms use the same code where possible. For all domains, we started with a uniform prior; that is, for every pair of states $s, s' \in S$ and

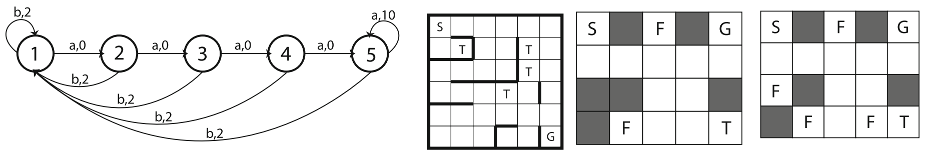

Fig. 1. Problem domains, from left to right: Chain, 6x6 maze, Dearden maze, larger maze

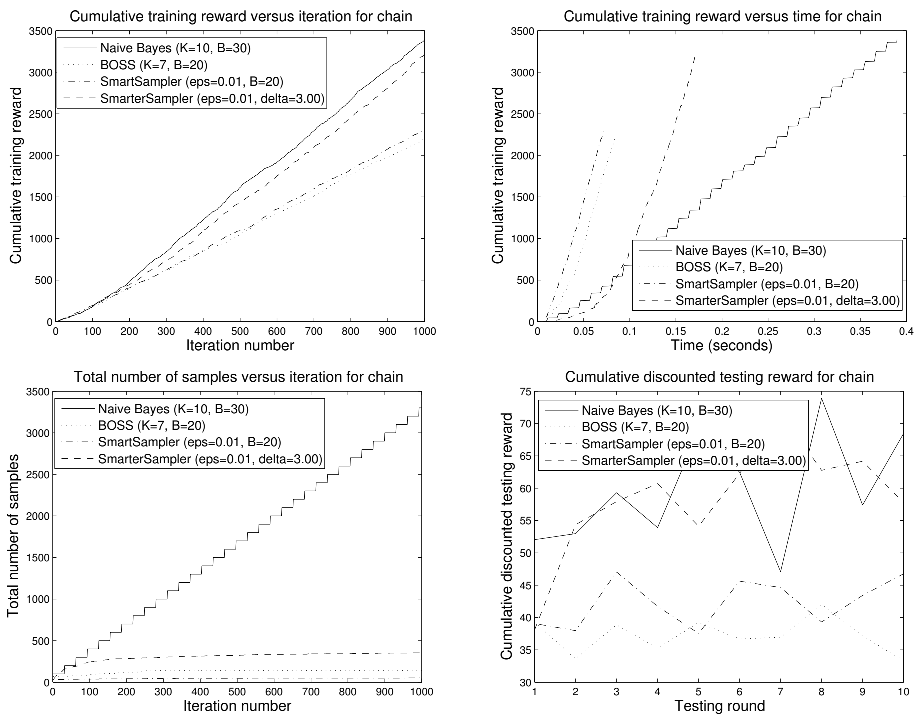

Fig. 2. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). Chain problem.

action a, the Dirichlet parameter was set to $\alpha_{s'}^{(s,a)} = 1/|S|$. All results are averaged over 10 runs.

We ran experiments on a number of domains shown in Figure 1. The chain domain from (Strens, 2000) has 5 states and 2 actions; the action has the desired effect with 0.8 probability and has the opposite effect with 0.2 probability; in the figure the transitions are labeled with the action name and the reward. The 6x6 maze domain from (Russell & Norvig, 1994) has 36 states and 4 actions (up, down, left, right); the agent starts at the cell marked 'S' and aims to reach the goal 'G', trying to avoid the trap states 'T'; the action moves the agent in the desired direction with probability 0.8, and the rest of the time the agent moves in a direction perpendicular to the desired direction, where moving into a wall will maintain the agent in the same state; each move has a cost of 0.001, and the agent receives a terminal reward of -1 or $+1$ for entering a trap or the goal state, respectively, whereupon the agent is placed back in the start state. The first maze domain is from (Dearden et al., 1999) and has 56 states; the agent moves in the desired direction with probability 0.9, and the rest of the time the agent moves in a direction perpendicular to the desired direction, where moving into a wall will maintain the agent in the same state; the agent receives a penalty of -10 for entering a trap state 'T' and upon entering the goal state, receives a terminal reward of 1 for every flag 'F'

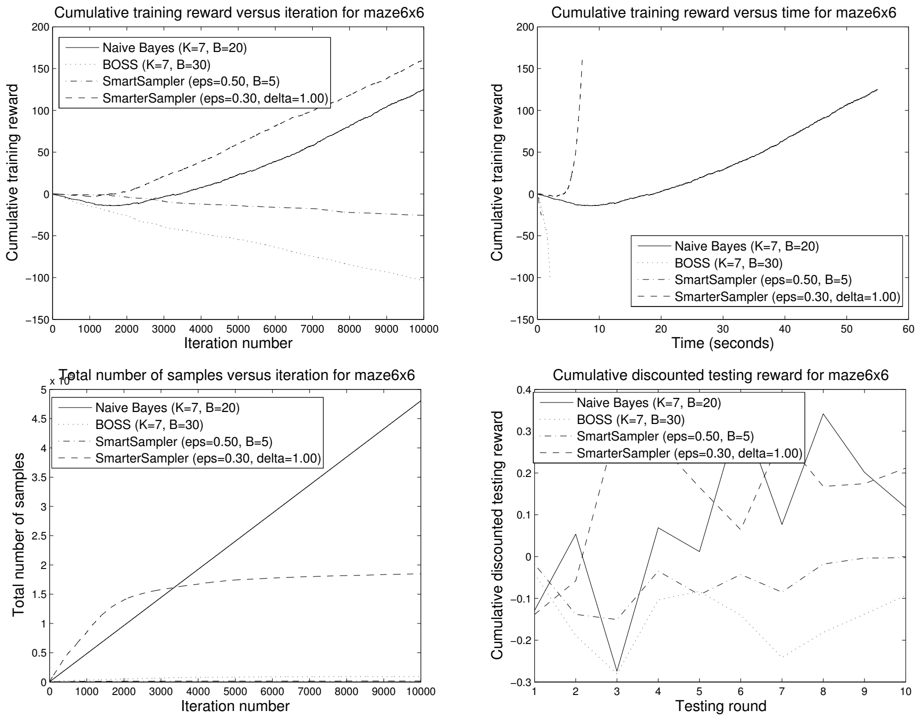

Fig. 3. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). 6x6 maze problem.

the agent has caught. The larger maze domain is similar, but with extra flags, which increases the state size to 240. We plot the cumulative reward received during learning versus iteration and the cumulative reward received throughout learning versus time (top 2 graphs in each figure). In previous algorithms, the number of transition models sampled for each state-action pair was the same. In our methods they vary, so we also plot the total number of transition models sampled versus iteration. Finally, we tested the policy at 10 evenly spaced intervals for 1000 iterations throughout the learning process and plot the sum of discounted rewards for each test episode. The results for the chain problem are illustrated in Figure 2; for the 6x6 maze in Figure 3; for the maze problem in Figure 4; and for the larger maze in Figure5.

We ran each of the algorithms with varying parameters and picked the ones that gave the best balance between training reward accumulation, testing reward accumulation and running time. We can see that both the SmartSampler and SmarterSampler have a clear advantage over BOSS in terms of return and time; although the Naive Bayes algorithm seems to generate fairly good returns, it is extremely slow compared to the other algorithms. In the larger domains, the advantage of using SmarterSampler becomes more evident. In both the 6x6 maze and the maze problem of (Dearden et al., 1999) we

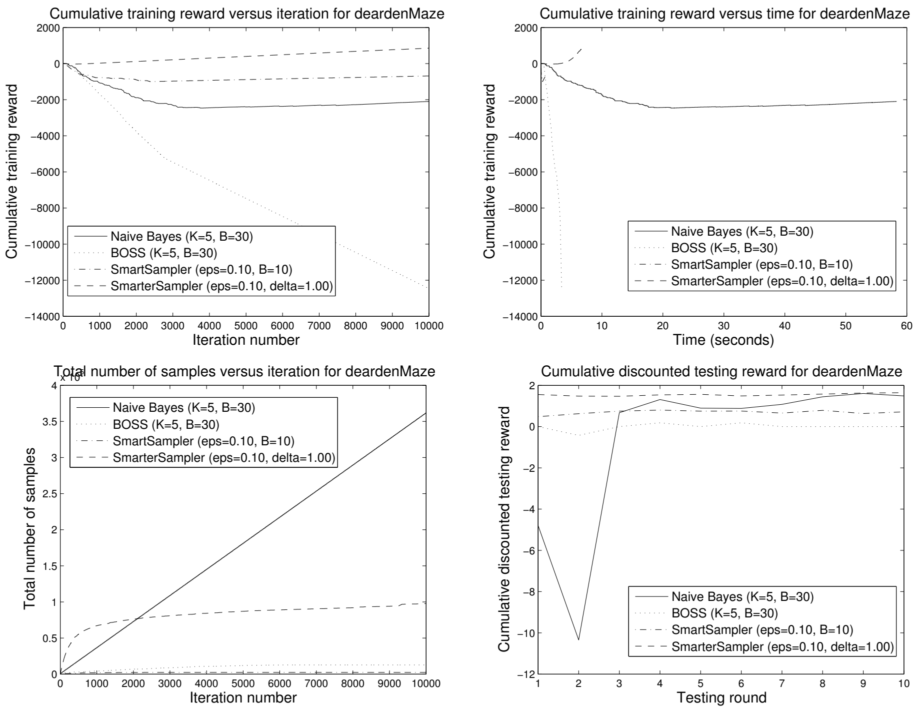

Fig. 4. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). Maze from (Dearden et al., 1999).

can see that SmarterSampler is performing the best exploration in the least amount of time; both BOSS and SmartSampler appear to get stuck on a "safe" policy with little returns. SmarterSampler is once again able to obtain higher returns both during training and during testing, with a reasonable running time.

We also plot the effect of varying the parameters of the SmarterSampler algorithm in Figures 6 and 7. As expected, there is a tradeoff between rewards obtained (both during training and during testing) and the running time of the algorithm. We can also see that the algorithm is not too sensitive to small changes in the parameters, which makes it a robust choice for model-based Bayesian exploration.

6 Discussion and Future Work

We presented an approach to Bayesian RL based on a bound on the difference between the value function of a model sampled from a Dirichlet distribution and the mean of that distribution. There has already been work on bounding the difference in value functions between two systems with equal state spaces but different dynamics. In (Müller, 1997), a generalized theory for bounding the difference in value function (up to a finite horizon) is presented, using various probability metrics (such as the Kantorovich metric) along with structural properties of the value function. The result is theoretically pleasing, but it is somewhat involved and does not provide an algorithm for computing the bounds. The paper by (Ferns et al., 2004) introduces bisimulation metrics for MDPs. These metrics are based on the Kantorovich probability metric, and in fact, can be viewed as a slightly modified instance of the bounds of (Müller, 1997). Other related papers performed sensitivity analysis of the effect of varying dynamics on a stochastic system, e.g. (Hinderer, 2005) and (Smith & McCardle, 2002). We relied on total variation in this paper because it is easy to compute (a big asset in the context of Bayesian RL) and still gives powerful results in the particular case of the Dirichlet distribution.

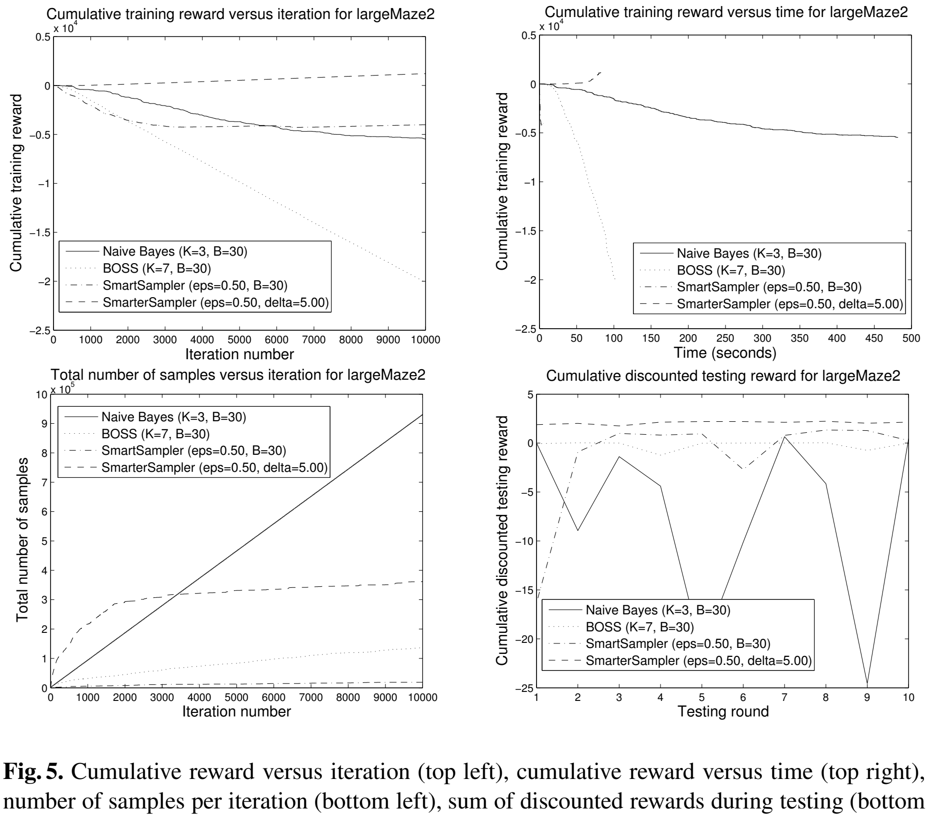

Fig. 5. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). Larger maze.

Although in (Asmuth et al., 2009) the authors demonstrated how to choose these values in order to guarantee near-optimal exploration with high probability, the values are very difficult to compute, and the authors resorted to choosing them manually in practice. We presented a method for choosing these values dynamically by using statistics

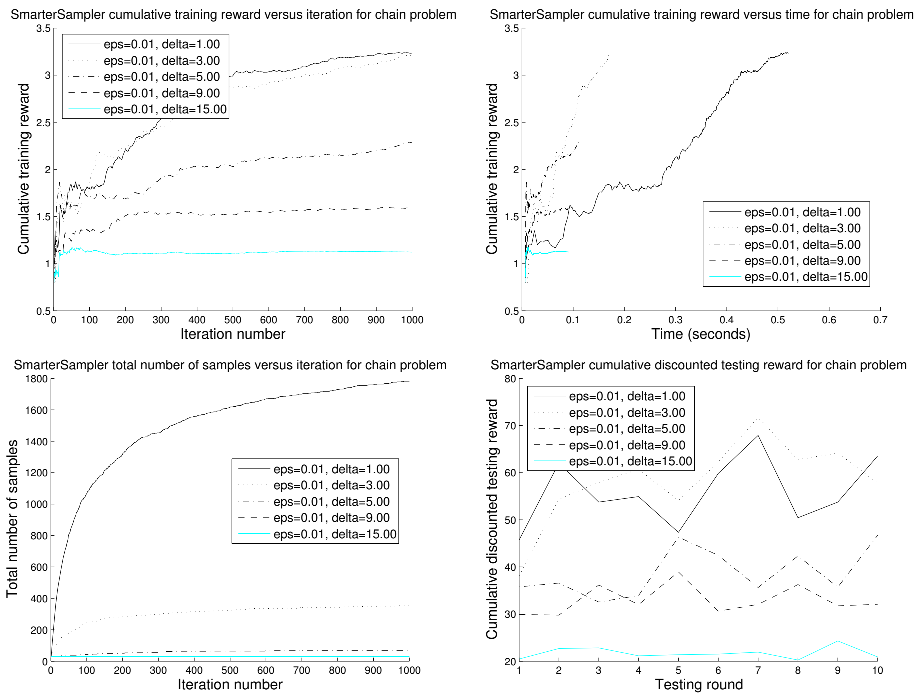

Fig. 6. SmarterSampler. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). Chain problem.

of the posterior distribution. again, computing these values is cheap and our experimental results demonstrate that they produce superior performance, both in terms of returns and running time. Although the UCRL algorithm from (Auer, 2000) is intended for the undiscounted reward case, our algorithms also have strong connections to it and it merits further investigation and an empirical comparison.

We motivated our algorithms by first providing theoretical bounds on the difference in value function between the mean model and a random sample from the posterior distribution over models, when this posterior is a Dirichlet distribution. Although we used the Dirichlet for our experimental results, our algorithm can hold for any other type of distribution, as long as we can compute the means and variances of the marginals of the mean model.

For simplicity, we assumed that the reward function was known beforehand, but our method can be adapted easily to the case when the rewards are not known. If we assume there is a finite set of possible rewards, we can also maintain a distribution over reward models and use similar techniques for determining when and how much to sample. This situation would be of particular interest when the rewards are stochastic.

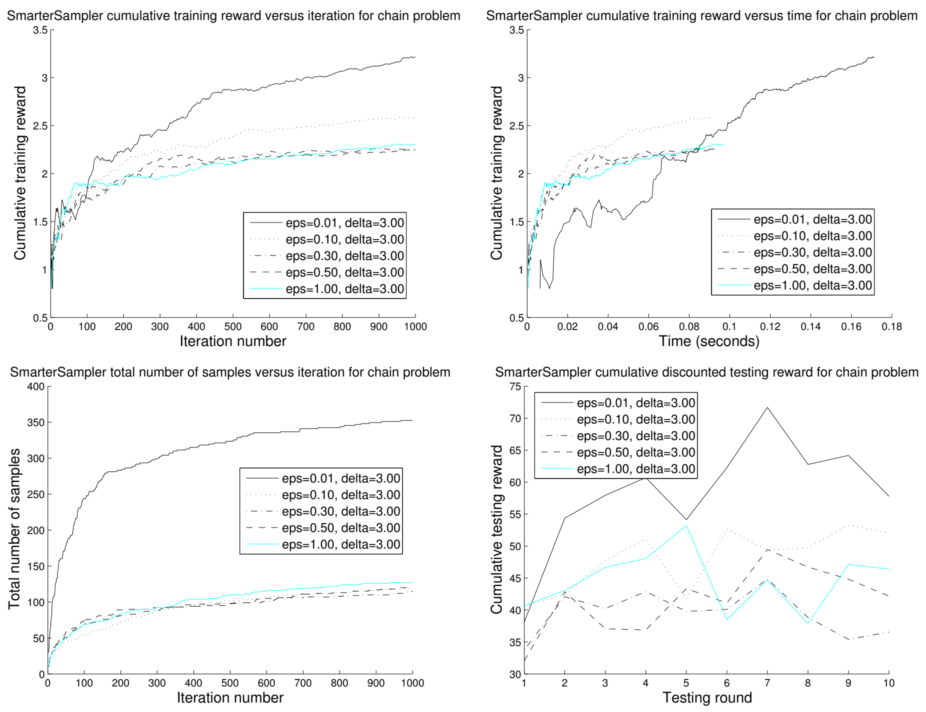

Fig. 7. SmarterSampler. Cumulative reward versus iteration (top left), cumulative reward versus time (top right), number of samples per iteration (bottom left), sum of discounted rewards during testing (bottom right). Chain problem.

Acknowledgements

This work was funded by NSERC and ONR. We would also like to thank the reviewers for their helpful comments.

References

Asmuth, J., Li, L., Littman, M.L., Nouri, A., Wingate, D.: A Bayesian Sampling Approach to Exploration in Reinforcement Learning. In: Proceedings of The 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009 (2009)

Auer, P.: Using upper confidence bounds for online learning. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (2000)

Dearden, R., Friedman, N., Andre, D.: Model based Bayesian exploration. In: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI 1999 (1999)

Duff, M.: Design for an optimal probe. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 131–138 (2003)

Engel, Y., Mannor, S., Meir, R.: Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 154–161 (2003)

Ferns, N., Panangaden, P., Precup, D.: Metrics for finite Markov decision processes. In: Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence, pp. 162–169 (2004)

Hinderer, K.: Lipschitz Continuity of Value Functions in Markovian Decision Processes. Mathematical Methods of Operations Research 62, 3–22 (2005)

Kolter, J.Z., Ng, A.Y.: Near-Bayesian Exploration in Polynomial Time. In: Proceedings of the Twenty-Sixth International Conference on Machine Learning (2009)

Müller, A.: How does the value function of a Markov Decision Process depend on the transition probabilities? Mathematics of Operations Research 22, 872–885 (1997)

Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 697–704 (2006)

Price, B., Boutilier, C.: A Bayesian approach to imitation in reinforcement learning. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI 2003 (2003)

Puterman, M.L.: Markov Decision Processes. John Wiley & Sons, New York (1994)

Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1994)

Smith, J.E., McCardle, K.F.: Structural Properties of Stochastic Dynamic Programs. Operations Research 50, 796–809 (2002)

Strens, M.: A Bayesian Framework for Reinforcement Learning. In: Proceedings of the 17th International Conference on Machine Learning, ICML 2000 (2000)

Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

Wang, T., Lizotte, D., Bowling, M., Schuurmans, D.: Bayesian sparse sampling for on-line reward optimization. In: Procedings of the 22nd International Conference on Machine Learning, ICML 2005 (2005)