Measures of Uncertainty for Policy Evaluation

Cosmin Paduraru COSMIN@CS.MCGILL.CA
Doina Precup DPRECUP@CS.MCGILL.CA
Mahdi Milani Fard MAHDI.MILANIFARD@MAIL.MCGILL.CA
School of Computer Science, McGill University, Montreal, Canada

Abstract

Estimated measures of uncertainty, such as confidence intervals, are a widely used tool in applied science. For sequential decision-making problems, however, there has been little work on computing such measures from data. For existing approaches, there has been little work evaluating their usefulness. In this paper, we review existing methods for measuring the uncertainty in estimating the expected return of a decision-making policy in a finite Markov Decision Process. We also present two new methods based on estimating returns for one policy based on data collected using a different policy. This scenario is especially important in medical applications, such as adaptive treatment design, in which randomized clinical trials are used to collect data, but the goal is to evaluate the outcomes of several different courses of treatment. Ideally, this data should allow statistical significance comparisons between different treatment policies. We illustrate the strengths and weaknesses of different uncertainty estimation method through a series of simulated problems, providing intuition for how accurate each method is and how tight its bounds are.

1. Introduction

Reinforcement learning is becoming a standard tool in an increasing range of scientific, medical and engineering applications, e.g. adaptive treatment design (Pineau et al, 2007), power system management (Glavic et al, 2006), computer network design and routing (Littman et al, 2004), etc. In such applications, one typically has access to a batch of data gathered ahead of time, but directly interacting with the system is very difficult, due to logistic constraints or safety concerns. Building a faithful simulation of the system may also be very difficult. Several *batch reinforcement learning* methods have been developed for this problem (e.g. Lin, 1992; Ernst et al., 2005; Riedmiller, 2005). Some empirical evidence (e.g. Kalyanakrishnan and Stone, 2007) suggests that batch methods lead to similar results as on-line methods, but using fewer samples.

In batch reinforcement learning, the data consists of a set of transitions (s, a, s', r) sampled by environment interaction. The policy according to which these were obtained is usually randomized, and it may be unknown to the user of the batch. The goal is to use the samples in order to obtain an optimal policy. However, in many of these applications, one may also need to provide a measure of uncertainty in the estimated value of the policy that was obtained. For example, in adaptive treatment design, in order for a treatment sequence to be adopted by the medical community, one may need to provide evidence that it is better than other options in a *statistically significant* way. For individual drug treatments, one can easily establish p-values as proof of significance; but how would this be done for a reinforcement learning method? In this paper our goal is to explore methods for providing uncertainty estimates for the value of a policy estimated using batch reinforcement learning. Two main approaches to this problem already exist in the literature. Mannor et al. (2007) establish variance bounds for the value function based on a second-order approximation; they also discuss a related Bayesian setting (we will discuss the details of their method later). Bayesian reinforcement learning methods can also provide uncertainty estimates; a typical approach is to maintain a probability distribution either over models (e.g. Dearden et al., 1999) or over value functions (e.g. Dearden et al., 1998, Engel et al., 2003). Uncertainty estimates can then be established by drawing

Preliminary work. Under review by the North East Student Colloquium on Artificial Intelligence (NESCAI) 2010. Do not distribute.

samples from this distribution. In this paper, we add two new approaches: a Monte Carlo (trajectory-based) approach, based on estimating returns from the existing batch of data, and a model-based approach, based on bootstrapping (Efron, 1979). We analyze the performance of these models in three different problem domains: a random walk, randomly generated MDPs, and a simple domain inspired by the adaptive treatment design problem. The empirical results show that useful uncertainty estimates can indeed be obtained in the batch scenario with reasonable amounts of data. However, the performance of the different methods is quite different. One of our main findings is that for reinforcement learning problems frequentist methods, which are unbiased, tend to perform better than Bayesian methods, which are biased by the choice of prior.

In the remainder of the paper we present necessary background and notation, review the methods that we will compare and present the empirical results, as well as a discussion.

2. Background

2.1. Confidence intervals and credible intervals

A $1-\alpha$ confidence interval for a parameter θ is an interval $C = (LB, UB)$, where LB and UB are functions of the data, such that

$$P(\theta \in C) \geq 1 - \alpha, \forall \theta \in \Theta.$$

The interpretation of confidence intervals is that if a large set of $1-\alpha$ confidence intervals are constructed for unrelated parameters using new data every time, then a fraction of at least $1-\alpha$ of the intervals will trap the true parameter value.

Let θ be the expected value of some random variable X. We will focus on the case when we cannot assume that X is normally distributed or that we have the exact value of X's variance. In this setting, a standard statistical approach for computing confidence intervals is to use the fact that the quantity:

$$T = \frac{\bar{X} - \theta}{S/\sqrt{n}}$$

has an approximate t distribution with $n-1$ degrees of freedom. Here, S^2 is the unbiased variance estimator computed from a sample of size n. This approximation works quite well for relatively small values of n (typically chosen to be above 30) and for most distributions of X. However, one should keep in mind that the only way to get reliable confidence intervals for general distributions is to use concentration inequalities such as the Hoeffding bound. Unfortunately, these general bounds are rather loose.

If the t distribution approximation is used, a $1-\alpha$ confidence interval around θ is given by:

$$[\bar{X} - t_{\alpha/2}(n-1)S/\sqrt{n}, \quad \bar{X} + t_{\alpha/2}(n-1)S/\sqrt{n}]$$

where $t_{\alpha/2}(n-1)$ is such that $P(T > t_{\alpha/2}(n-1)) = \alpha/2$.

In Bayesian statistics, uncertainty is usually captured by "credible intervals", defined to be intervals including no less than a $1-\alpha$ fraction of the mass of the posterior distribution. With the Bayesian assumption of the correctness of the prior, one can guarantee that such credible interval will include the true parameter value (taken to be a random variable from the Bayesian perspective) with probability no less than $1-\alpha$. In most of the interesting cases in machine learning, if the priors are not subjective, credible intervals and confidence intervals coincide.

2.2. Markov Decision Processes and Policy Evaluation

A Markov Decision Process (MDP) is a tuple $M = \langle S, A, P, R, \gamma \rangle$, where S is the set of states, A is a set of actions, P is the transition probability distribution, with $P(s'|s,a) = P(s_{t+1} = s'|s_t = s, a_t = a)$ denoting the probability of the transition from s to s' when taking action a, $R : S \times A \to \mathbb{R}$, $R(s,a) = E[r_{t+1}|s_t = s, a_t = a]$ is the reward function and $\gamma \in (0,1]$ is a discount factor. The agent interacts with the MDP via a policy $\pi : S \to A$. For the moment, we consider finite state and action spaces.

The value function of a policy π in a state s is defined as $V^\pi(s) = E[CR^\pi(s)]$, where

$$CR^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

called the "cumulative reward" or "return", is the sum of future discounted returns obtained after starting in $s_0 = s$. Since the value function is the expectation of a random variable, frequentist confidence intervals can be used for evaluating the uncertainty in estimating it. In addition, Bayesian methods can be used for computing credible intervals by maintaing an approximate posterior over the value function.

3. Computing confidence intervals for the value function of a given policy

3.1. Existing methods

There are several existing approaches to computing confidence intervals around the value function for a specific policy. Kaelbling (1993) used the central limit theorem to compute confidence intervals for the action-conditioned expected reward in bandit problems. These confidence intervals were used to guide exploration, by choosing the action with the highest upper bound. Kaelbling also proposed a heuristic algorithm for exploration in sequential problems, where the confidence intervals were computed by using the right hand side of Q-learning updates as samples of the return. Meauleau and Bourgine (1999) extend Kaelbling's method to take into account the uncertainty at future states. One should keep in mind that these are heuristic algorithms, making no claims that they compute correct confidence intervals for the value function in sequential problems.

Mannor et al. (2007) provide a closed-form approximation to the bias and variance of value function estimation from a finite data set. They start from the value function estimate:

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{R}^\pi = \left(\sum_{i=0}^{\infty} (\gamma \mathbf{P}^\pi)^i \right) \mathbf{R}^\pi \qquad (1)$$

where $\mathbf{V}^\pi \in \mathbb{R}^{|S|}$ contains the value estimates for all states, $\mathbf{P}^\pi \in \mathbb{R}^{|S| \times |S|}$ is the matrix of transition probabilities between states under π and $\mathbf{R}^\pi \in \mathbb{R}^{|S|}$ is the vector of expected rewards under π. Then they consider the estimates of $\mathbf{R}^\pi$ and $\mathbf{P}^\pi$ as a sum of a mean and an error term, which depend on the number of samples used for the estimation. They expand the equation above using these estimates and truncate it to include only second-order terms, which are used to compute the bound. For details, we refer the reader to their paper.

Measuring the uncertainty around the value function has also been addressed in the Bayesian reinforcement learning literature. In this context, uncertainty is expressed via the posterior probability distribution over the value function. In the model-based setting, the posterior over value functions can be approximated by sampling from the posterior over models, which in turn is easy to update if Dirichlet priors are used (e.g. Dearden et al., 1999). A similar approach is used by Jong and Stone (2005) and Tetreault et al. (2007), who are explicitly interested in computing credible intervals for the value function. In the model-free setting, Dearden et al. (1998) compute the posterior over value functions analytically for the case when the returns are normally distributed and the value functions at different states are independent. The Gaussian process temporal difference (GPTD) work (e.g. Engel et al., 2003) relaxes the independence assumption, while keeping the normality assumption. Note that GPTD is designed for use in continuous state and action spaces via value function kernels.

generate-sample-returns(π, batch of transitions)
    for each (s, a) pair in the batch of transitions
        initialize transList(s, a) to be an empty queue
    for each (s, a) pair in the batch of transitions
        for each (s, a, s', r) in the batch of transitions
            add (s', r) to transList(s, a)
    $s \leftarrow s_0$; $i \leftarrow 1$; $cr_1 = 0$
    repeat
        sample $a \sim \pi(s)$
        dequeue (s', r) from transList(s, a)
        if transList(s, a) is empty return $cr_1, ..cr_{i-1}$
        $CR_i \leftarrow CR_i + \gamma r$
        if s' is terminal $s \leftarrow s_0$; $i \leftarrow i + 1$; $cr_{i+1} = 0$
        else $s \leftarrow s'$

Table 1. Algorithm for generating i.i.d. samples of $CR^\pi(s_0)$ from a batch of sampled transitions in the episodic undiscounted case.

3.2. Trajectory-based approach

If the agent was able to gather data on-line, it Monte Carlo return samples $cr_1, ...cr_n$ of $CR^\pi(s_0)$ could be used to compute confidence intervals using standard statistical methods. Here we present a method that can be used when on-policy interaction is not possible. Our method works by using the sampled transitions available in the batch to generate i.i.d. (independent and identically distributed) samples of the return $CR^\pi(s_0)$; these samples can then be used as if they were collected on-policy.

The algorithm for generating i.i.d. samples of $CR^\pi(s_0)$ is outlined in Table 1. It works by piecing together the transitions stored in the batch into sample trajectories for π. For each state-action pair (s, a) contained in the batch, a list of available samples is maintained. The samples of $CR^\pi(s_0)$ are then generated by taking transitions from this list, instead of querying the environment as would be done in the on-policy setting. Initially, the list of transition samples for (s, a) consists of all states and rewards that were observed to follow (s, a) in the batch; however, every transition sample that is returned following a query for (s, a) is deleted from (s, a)'s list. The algorithm stops when a query cannot be satisfied at some state; this means that no

more samples of $CR^\pi(s_0)$ can be generated. Note that the algorithm can only compute confidence intervals at states from which enough simulated trajectories are available.

Theorem 1. *The values $cr_1, ...cr_n$ returned by the algorithm from Table 1 are i.i.d. samples of the on-policy return $CR^\pi(s_0)$.*

Proof. A valid procedure for obtaining n independent samples from a random variable X is to generate m samples of X, where $m \geq n$, and then take the first n out of those m samples. Therefore, given a state s and an action a, a valid procedure for generating n independent samples of the next state according to the state transition model P and n independent samples from the reward according to the reward model R is to collect a batch of m samples of the next states and rewards, where $m \geq n$, and then take the first n of these transitions. We will refer to this property as *transition sampling equivalence*.

The on-policy procedure for sampling from $CR^\pi(s_0)$ is the following: starting in $s = s_0$, initialize i to 1 and cr_i to 0, sample an action a according to $\pi(s)$, then sample a next state s' and a reward r according to the state transition P and the reward model R respectively, add r to cr_i, set $s = s'$, and continue. Every time the terminal state is reached, a new sample of $CR^\pi(s_0)$ is obtained, s is reset to s_0, i is incremented by 1 and cr_i is initialized to zero. This procedure can generate new independent samples of $CR^\pi(s_0)$ as long as individual transitions can be sampled. If for some reason the action, next state or reward cannot be sampled, the procedure stops.

The on-line procedure outlined above performs exactly the same steps as the outlined algorithm. The only difference is the way the samples of s' and r are generated: for on-policy evaluation the environment is queried every time the agent is in state s and has sampled action a, whereas in our case the samples are collected in advance. However, because of transition sampling equivalence both methods generate independent transition samples. Thus, running the algorithm produces independent samples of $CR^\pi(s_0)$. □

3.3. Bootstrap estimate

The idea of this method is to use bootstrap estimation (Efron, 1979) to construct confidence interval estimates. More precisely, we will create replicates of the batch of data by re-sampling with replacement. Each replicate is the same size as the original data set. Each new batch is used to compute a model, and this model is used to estimate the value function (as described in Equation 1). This approach is simple and general, as it does not rely on any assumption about the value function.

3.4. Strengths and weaknesses of different methods

The algorithm presented in Table 1 produces samples from the true distribution of returns, without any additional assumptions. In order to compute confidence intervals based on these returns, however, we have to assume that the empirical mean is distributed according to a normal or t distribution for a large enough number of samples, a standard assumption in applied statistics.

Bayesian model-based estimation (Dearden et al, 1999; Tetreault et al., 2007) samples models from the Dirichlet posterior and then computes the value function for those models, approximating the posterior distribution via samples. As more models are sampled from the posterior, this approximation should become more accurate. Note that this method produces credible intervals, not confidence intervals.

If prior information is available in distributional form, then Bayesian methods provide the ideal way of incorporating this information. However, because they also *require* that a prior be provided, they will always be subject to bias. This is widely recognized and accepted in the statistics community. However, to date there has not been much discussion of how the bias in the model estimation affects the value estimates in model-based reinforcement learning. In model-based Bayesian reinforcement learning, the posterior over value functions is computed from the posterior over models. This involves a higly non-linear transformation of a distribution, and as such it can amplify whatever bias existed in the original distribution. This means that the Bayesian credible intervals cannot guarantee that the true parameter is within the interval a certain percentage of the time as the experiment is repeated, even if a relatively benign prior is being used.

The analytical approximation of Mannor et al. (2007) involves several potential sources of error. First, they use a second-order approximation for computing the variance. Second, estimates of the transition and reward model have to be used instead of the true model in the formulas. Third, in order to use the variance estimates for computing confidence intervals we need to make the same kind of asymptotic normality assumptions as for the trajectory-based method.

Bootstrapping in general is asymptotically consistent,

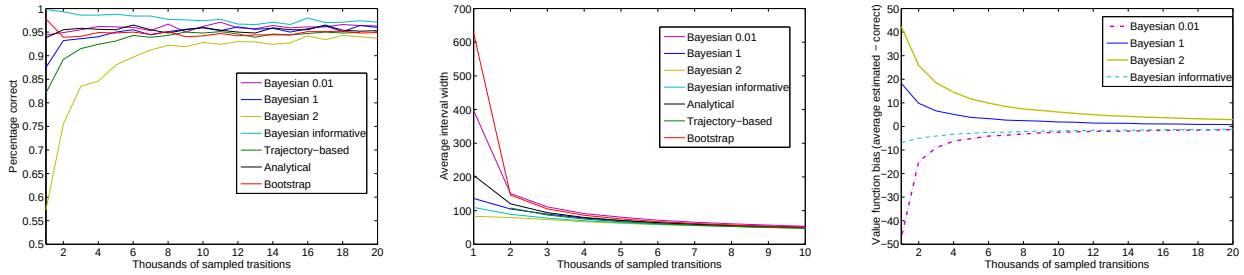

Figure 1. Results for the chain MDP: percentage of time the confidence intervals capture the true value (left); widths of the confidence intervals (middle); bias of the Bayesian methods (right)

but there are no theoretical guarantees in the finite sample case, and these estimates can be biased (Shenkar, 1985).

We expect the trajectory-based and model-based methods to differ based on the type of problem at hand. For example, if all trajectories get the same cumulative reward but pass through different states, the trajectory-based solution will produce very tight confidence intervals whereas a model-based approach will produce wider intervals due to the uncertainty in estimating state transition probabilities. On the other hand, suppose there are enough transitions to estimate the model well for all but one state, and this state happens to be visited frequently by the policy we are evaluating. In this situation the trajectory-based method can only use a small number of trajectories and will discard the rest of the data, whereas model-based methods can achieve better confidence by using all the data that is available.

In light of the above considerations, it seems unlikely that we can find a method that is theoretically superior to the other methods. Instead, we will investigate the performance of these methods empirically

4. Empirical comparison

4.1. Experimental setup

We compare all the methods outlined above. As described, the trajectory-based algorithm applies to episodic tasks. In non-episodic tasks, we will assume that we are interested in the value of some starting state s_0. We generate returns by starting in s_0 and summing the sampled rewards for n steps, where n is the smallest number for which γ^n is smaller than a threshold, chosen to be 0.0001 in our experiments. In all of the domains the rewards are fixed and available to the agent. This setup was chosen for simplicity and because rewards are provided by the user in many realistic situations. Moreover, variance of the rewards has a comparatively smaller effect than the variance in the probability estimates, because rewards affect the value function linearly.

For all methods, we measure the width of 95% confidence intervals and the percentage of time the intervals capture the correct value function, when the experiment is repeated. Note that this is a typically frequentist evaluation methodology. While we acknowledge that Bayesian methods will be correct with respect to their prior, we are still interested in evaluating how much their performance is affected when the prior is not correct. This is especially important since there is little available analysis of the effect of the prior distribution in model-based reinforcement learning.

In order to evaluate the percentage of intervals that capture the correct value, we generate m different data sets for each problem using a fixed behavior policy b. We use each of these data sets to compute confidence or credible intervals for some target policy π; then we compute the percentage of these intervals that contain the correct value. The widths are also averaged over the m data sets. In all our experiments, we used $m = 1000$.

4.2. Markov Chain

The first problem is a simple Markov Chain with five states.The leftmost state is the starting state and the rightmost state is the terminal state. The probability of transitioning to the state immediately to the right is 0.3, and the probability of transitioning to the state immediately to the left is 0.7 (in the leftmost state, there is a self-loop with prboability 0.7). The rewards are -1 per time step. The discount factor is $\gamma = 1$. The correct value function of the starting state for this chain is -115.31. We generated data sets ranging in size from 1000 to 20000 transitions.

Our main goal in this task is to analyze the effect of the Bayesian prior on the credible intervals produced by Bayesian sampling. We use four different Dirichlet priors for the model. The first one is an informative prior, assigning initial counts of 70 for the transitions that occur with probability 0.7, counts of 30 for the

Table 2. Percentage of intervals containing the value function for the random MDPs

Index	1	2	3	4	5	6	7	8	9	10	Average
Analytical	0.97	0.97	0.96	0.98	0.97	0.97	0.97	0.96	0.97	0.98	0.9732
Trajectory	0.94	0.95	0.95	0.96	0.96	0.95	0.96	0.96	0.95	0.95	0.9527
Bayesian	0.90	0.90	0.97	0.82	0.95	0.93	0.75	0.86	0.95	0.96	0.8977

Table 3. Average interval width for the random MDPs

Index	1	2	3	4	5	6	7	8	9	10	Average
Analytical	0.0145	0.030	0.044	0.023	0.090	0.025	0.038	0.043	0.025	0.019	0.0353
Trajectory	0.189	0.135	0.254	0.204	0.336	0.256	0.193	0.227	0.325	0.202	0.2321
Bayesian	0.014	0.029	0.044	0.023	0.080	0.025	0.037	0.042	0.028	0.019	0.0342

transitions that occur with probability 0.3, and counts of 0.001 for all other transitions. The second one is an agnostic prior, assigning each transition a prior count of 0.001. The third and fourth ones are weak uniform priors, assigning each transition counts of 1 and 2, respectively.

The results are presented in Figure 1. The left panel shows the percentage of time that the confidence interval traps the correct value. The middle panel shows the average size of the confidence interval, and the right panel shows the bias for all of the Bayesian methods (the other methods are unbiased). As expected, the informative prior produces good confidence intervals; however, the uninformed priors have a very strong bias, and as a result do not perform well. Intuitively, with the uniform prior, the value function is overestimated, because the model makes the transition to the reward state too likely, even from states where this transition is impossible. A similar effect should be expected, for example, in goal-based tasks where uniform priors are used. Note in particular the Bayesian methods with prior counts of 2, which produces tight confidence intervals around the wrong value. The bias is particularly strong with fewer samples, which is exactly the case for which Bayesian methods are said to be preferred. The weak prior also exhibits a bias, though in the other direction. The frequentist methods all perform well, though the analytical method seems to produce the tightest intervals with small amounts of data.

4.3. Random MDPs

The performance of the methods in non-episodic tasks was tested on a sequence of ten randomly generated MDPs. Each MDP had 20 states and two actions, uniform rewards in $[0, 1]$, and a discount factor of $\gamma = 0.9$. For each MDP, each of the two actions took the agent to one of three randomly selected next states, with the probabilities of transitioning to each of these three states chosen uniformly randomly. The behavior policy selected action 1 with probability 0.2 and action 2 with probability 0.8 in each state. For the target policy, the probabilities were 0.6 and 0.4, respectively. We generated data sets of $100,000$ transitions each from these MDPs. For the Bayesian sampling method a weak uniform Dirichlet prior was used with prior counts of 1. Results are shown in Tables 2 and 3.

As expected, the trajectory-based method is underperforming the other methods in this domain. Recall that the trajectory-based approach needs to sample transitions without replacement until γ^n falls below some threshold. This leads to a large number of sample transitions being required to obtain a sample of the return, and thus to wide confidence intervals. As we decrease the discount factor, the gap between the trajectory-based methods and the model-based methods decreases (results not shown here).

4.4. Simulated clinical trial

We simulated a multi-stage clinical trial with two stages and three possible treatments at each stage (the actions). We used three state variables: a patient's state of health, which can have one of two values (bad or mediocre), the previous treatment administered, and the stage of the trial. The dynamics are as follows: all patients enter the trial in the "bad" state of health. From there, they can transition after the first stage to staying bad or going to mediocre. For treatment 1, the probability of getting better after the first stage is 0.25, while for the other two treatments it is 0.2. Once in the second stage, patients can remit (cure) if their level of health is "mediocre". This happens with probability 0.3 for all treatments, unless treatment 3 was taken at the first stage and treatment

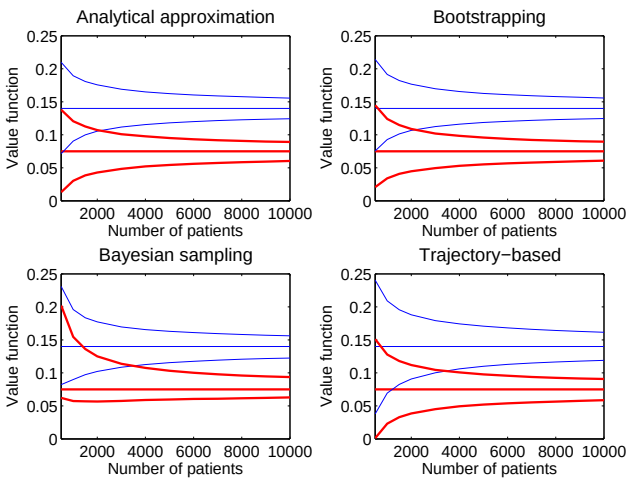

Figure 2. Clinical domain

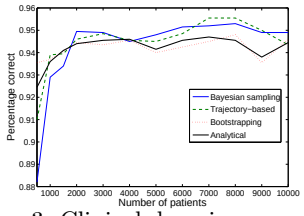

Figure 3. Clinical domain percentages

2 at the second stage, in which case the probability of remittance is 0.7. These dynamics are meant to illustrate the sequential aspects of multi-stage clinical trials, where the treatment that looks best at the first stage may not be the best way to start a long course of treatment. The reward is 1 for remittance and 0 for all other transitions. The problem is episodic, and episodes end after the second stage, when the patient either remits or does not remit. There is no discounting. Thus, the value function can be interpreted as the probability of eventually remitting after starting in the "bad" state.

The behavior policy was to select a random treatment at each state. We used two different target policies: the first one selects treatment 1 at the first stage and treatment 2 at the second stage, while the second one selects treatment 3 at the first stage and treatment 2 at the second stage. The value function of these policies were 0.075 and 0.14, respectively.

Figure 2 shows the correct value function of the two policies, and the confidence intervals around it, established using the different methods. The Bayesian method (with prior counts of 1) again shows bias, while the other methods do not. With the frequentist methods, based on these graphics, a clinical trial with at least 2000 patients could yield statistical significance results differentiating the policies. Figure 3 shows the percentage of confidence intervals correctly trapping the value function. Bootstrap and the analytical method are very similar. The trajectory method seems to have a slight advantage at large numbers of samples.

5. Discussion and future work

Based on our findings, the frequentist methods can all be used successfully to provide confidence intervals for policy evaluation in batch reinforcement learning. The Bayesian model-based method works well with an informative prior, but more uninformed prirs lead to bias in the model, which gets amplified in the estimate of the value function. More empirical work would be useful in fully understanding the strengths and weaknesses of methods for computing uncertainty estimates. In particular, we would like to know how the methods scale with problem size, or whether there is some other feature of an MDP that is important for predicting the performance of these methods. Providing a similar analysis for continuous state spaces is obviously both important and challenging. To our knowledge, the only existing piece of work on constructing uncertainty measures for policy evaluation in continuous state spaces is GPTD (Engel et al., 2003). The work of Fonteneau et al. (2009), although only dealing with deterministic systems, is also highly relevant and could serve as a starting point for developing methods for stochastic systems.

References

Dearden, R., Friedman, N., & Andre, D. (1999). Model based bayesian exploration. *Proceedings of UAI.*

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 761–768).

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*, 126.

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets Bellman: The Gaussian process approach to temporal difference learning. *Proceedings of ICML* (pp. 154–161).

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

Fonteneau, R., Murphy, S., Wehenkel, L., & Ernst, D. (2009). Inferring bounds on the performance of a control policy from a sample of trajectories. *Proceedings of ADPRL.*

Jong, N., & Stone, P. (2005). State abstraction discovery from irrelevant state variables. *Proceedings of IJCAI.*

Kaelbling, L. (1993). *Learning in embedded systems.* MIT Press.

Kalyanakrishnan, S., & Stone, P. (2007). Batch reinforcement learning in a complex domain. *Proceedings of AAMAS* (pp. 650–657). ACM.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*, 293321.

Littman, M. L., Ravi, N., Fenson, E., & Howard, R. (2004). Reinforcement learning for autonomic network repair. *ICAC* (pp. 284–285). IEEE Computer Society.

M. Glavic, L. Wehenkel, D. E. Damping control by fusion of reinforcement learning and control lyapunov functions. *Proceedings of The 38th North American Power Symposium* (pp. 361–367).

Mannor, S., Simester, D., Sun, P., & Tsitsiklis, J. (2007). Biases and variance in value function estimates. *Management Science*, *53*.

Meuleau, N., & Bourgine, P. (1999). Exploration of multi-state environments: Local measures and backpropagation of uncertainty. *Machine Learning*, *35*, 117154.

Pineau, J., Gendron-Bellemare, M., Rush, A., Ghizaru, A., & Murphy, S. (2007). Constructing evidence-based treatment strategies using methods from computer science. *Drug and Alcohol Dependence*, *88S*, S52–S60.

Riedmiller, M. (2005). Neural fitted Q-iteration - first experiences with a data efficient neural reinforcement learning method. *Proceedings of ECML* (p. 317328). Springer.

Shenker, N. (1985). Qualms about bootstrap confidence intervals. *Journal of the American Statistical Association*, *80*, 360–361.

Tetreault, J., Bohus, D., & Litman, D. (2007). Estimating the reliability of MDP policies: a confidence interval approach. *HLT-NAACL* (pp. 276–283).