

Propagating Uncertainty in POMDP Value Iteration with Gaussian Processes

Gatsby Unit Technical Report

Eric Tuttle and Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square, London WC1N 3AR, UK
{eric,zoubin}@gatsby.ucl.ac.uk

November 30, 2004

Abstract

In this paper, we describe the general approach of trying to solve Partially Observable Markov Decision Processes with approximate value iteration. Methods based on this approach have shown promise for tackling larger problems where exact methods are doomed, but we explain how most of them suffer from the fundamental problem of ignoring information about the uncertainty of their estimates. We then suggest a new method for value iteration which uses Gaussian processes to form a Bayesian representation of the uncertain POMDP value function. We evaluate this method on several standard POMDPs and obtain promising results.

1 Introduction

The problem of how to make decisions and plan strategies under uncertainty is of tremendous importance to areas of machine learning, operations research, management science, and many other fields. One general model for such problems which allows for uncertainty in the state of the world as well as in the effects of decisions is the Partially Observable Markov Decision Process (POMDP). While the POMDP model is attractive because of its more realistic uncertainty assumptions compared to Markov Decision Processes (MDPs), it has stubbornly resisted efficient solution algorithms for problems of any size to be of practical interest. This is true both for exact and approximate algorithms.

A wide variety of approximate algorithms have been proposed to find near-optimal policies in POMDPs. While an exact algorithm can build a solution secure in the knowledge that at each stage every bit of information is 100%

correct, once approximations are introduced errors are likely to be made and this confidence is no longer merited. A problem we have with many of the existing approximate algorithms for POMDPs is that they do not attempt to account for the varying amounts of inaccuracy and uncertainty of the information with which they construct a solution. Some of the bits of information used are more likely to be correct than others, and in that case any algorithm which ignores this is likely not to perform as well as it could.

It is this motivation which leads us to introduce a new algorithm combining approximate value iteration and Gaussian processes as a Bayesian representation of the uncertain value function. This enables the algorithm to place more weight on approximations it believes are good and to avoid being misled by mistakes. We will discuss this and other advantages of the Gaussian Process representation further in Section 3. But first we must introduce the central ideas.

2 Overview of the framework

Our discussion and algorithm depend on two frameworks, one for planning and one for regression. We give a brief overview of each in turn.

2.1 POMDPs

The POMDP model is a generalized form of the MDP model and can be described by a tuple $\{S, A, T, R, \Omega, O\}$. The first four objects are defined just as in an MDP: S is a (finite) set of states of the world, A is a (finite) set of actions which the agent has available to it, $T(s', s, a)$, $T : S \times A \rightarrow S$ defines the transitions between states, specifying for each possible action a the probability of moving from state s to state s' , and $R(s, a)$, $R : S \times A \rightarrow \mathbb{R}$ gives the immediate reward (r) the agent receives for taking action a in state s . The POMDP framework adds to the MDP Ω , a (finite) set of observations which the agent can perceive in the world, and $O(s', o, a)$, $O : S \times A \rightarrow \Omega$, which specifies the probability that the agent will perceive observation o after taking action a and arriving in state s' . As in MDPs, we often also specify a discount factor γ which indicates the amount by which the agent should devalue a reward one time step in the future.

The goal of the agent in a POMDP is to maximize its expected total sum of future rewards $E[\sum_{t=0}^T \gamma^t r_t]$ where T is the number of time steps left to go in a finite-horizon problem, or ∞ in an infinite-horizon problem. Finding a policy which achieves this goal is referred to as solving the POMDP. Unlike in an MDP, however, the agent in a POMDP is not assumed to have knowledge of the state; it only perceives the world noisily through observations as defined by O . Thus, even though the underlying world behaves as an MDP, the agent cannot form an optimal policy based upon the state alone. It must keep a complete history of its actions and observations, or a sufficient statistic, in order to act optimally. One such sufficient statistic is the belief state b , which is a vector of length $|S|$ whose elements $b_i = b(s_i)$ specify the agent's belief that it is in state s_i .

After taking an action a and seeing observation o , the agent updates its belief state using Bayes' Rule:

$$b'(s') = P(s'|a, o, b) = \frac{O(s', o, a) \sum_s T(s', s, a) b(s)}{P(o|a, b)}$$

The denominator, $P(o|a, b)$, is a normalizing constant and is given by the sum of the numerator over all values of s' . We refer to the function which returns b' as the state estimator $\text{SE}(b, a, o)$.

It turns out we can transform a POMDP into a “belief state MDP” [1]. Under this transformation, the belief states b become the (continuous) states of an MDP. The actions of the belief MDP are the same as the original POMDP, but the transition and reward functions are transformed to yield the following form of Bellman's equation for belief MDPs:

$$V^*(b) = \max_a \left[\sum_s b(s) R(s, a) + \gamma \sum_o P(o|a, b) V^*(\text{SE}(b, a, o)) \right] \quad (1)$$

As in any MDP, the optimal policy which the agent is trying to learn is the one which is greedy with respect to this optimal value function. The problem is that there are an infinite number of the continuous belief states, so solving this equation is hopeless. Value iteration, in which we compute V^* in the limit by iteratively applying the updates in (1) to an initial choice for V , also seems at first glance hopeless since the updates at each step would have to be computed over all of the continuous simplex of beliefs.

Exact solutions for the finite-horizon case take advantage of the fact that value functions for finite-horizon belief MDPs are piecewise-linear convex functions, and thus can be represented by a finite number of hyperplanes in belief-value space. Value iteration updates can be performed directly on the set of hyperplanes. In the infinite-horizon case, there can be infinitely many linear pieces to the convex value function, but we can get arbitrarily close to the true value function by taking the horizon far enough and using value iteration.

Unfortunately, value iteration is intractable for most POMDP problems with more than a few states. The size of the set of hyperplanes defining the value function can grow exponentially (in $|\Omega|$) with each step. Many attempts have been made to increase the efficiency of Sondik's original algorithm by pruning the set of hyperplanes. Identifying hyperplanes necessary to the exact solution also turns out to be intractable. Finding the exact optimal solution for a finite-horizon problem has been shown to be PSPACE-hard, and the story for the infinite horizon is worse (reviewed in [2]). This has led many researchers to examine approximate methods, one important class of which is value function approximation. In these methods, at each step of value iteration the true value function V is replaced by an approximate one, \tilde{V} .

In this paper we will not be working directly with the value function, but with the action-value function $Q(s, a)$ (also called the Q-function), which is

given by:

$$Q(b, a) = \sum_s b(s)R(s, a) + \gamma \sum_o P(o|a, b) \max_{\alpha} [Q(\text{SE}(b, a, o), \alpha)]$$

The Q-function $Q(b, a)$ gives the value of taking action a in state b and thereafter following the policy which is greedy with respect to V . It is related to the value function by $V(b) = \max_a Q(b, a)$.

2.2 Gaussian processes

Gaussian processes have received a great deal of attention as a method of performing Bayesian regression. A Gaussian process regressor defines a distribution over possible functions that could fit the data. In particular, the distribution of a function $y(\mathbf{x})$ is a Gaussian process if the probability density $p(y(\mathbf{x}_1), \dots, y(\mathbf{x}_N))$ for any finite set of points $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is a (multivariate) Gaussian. (See [3] for an introduction to Gaussian processes).

Typically, the mean of this process is taken to be 0 and our prior knowledge about the function is encoded in our choice of covariance function $K(\mathbf{x}_i, \mathbf{x}_j)$. This function can have any form so long as it is guaranteed to produce a positive semi-definite covariance matrix for any choice of points $\{\mathbf{x}_i, \mathbf{x}_j\}$.

Suppose we have observed a set of (training) points and target function values $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$. We believe the underlying function $y(\mathbf{x})$ to be distributed as a Gaussian process with covariance function K , and that the target values \mathbf{t} are related to the true function by the addition of Gaussian noise. Then,

$$\mathbf{t} = \mathbf{y} + \eta \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$$

where $\eta \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is the noise term and $\mathbf{C} = \mathbf{K} + \Sigma$, with $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$

Suppose we then wish to know the value of the function at some new (test) points $\{\mathbf{x}'_m\}_{m=1}^M$. Since the joint distribution of the function at all of the points is Gaussian, the conditionals are also Gaussian and given by:

$$\mathbf{y}(\mathbf{x}') | \mathcal{D} \sim \mathcal{N}(\mathbf{K}(\mathbf{x}, \mathbf{x}')^T \mathbf{C}^{-1} \mathbf{t}, \mathbf{K}(\mathbf{x}', \mathbf{x}') - \mathbf{K}(\mathbf{x}, \mathbf{x}')^T \mathbf{C}^{-1} \mathbf{K}(\mathbf{x}, \mathbf{x}'))$$

where $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ is a matrix of covariance between training and test points, and $\mathbf{K}(\mathbf{x}', \mathbf{x}')$ is a matrix of covariance between the test points and themselves. Thus, if we need a prediction for the function value at a test point we can use the mean as the most probable choice, and we also have information about how certain we are about that value in the form of the (co-)variances of the function values. The computational cost of making a prediction at M test points based upon N observed points scales as $\mathcal{O}(N^3 + MN)$, the first term being the cost of inverting an $N \times N$ matrix. As differentiation is a linear operator, it is also straightforward to compute the mean and variance of any derivatives of the function, which also have Gaussian distributions. And if derivative values are known for points in the training set, these can also be incorporated as observed data [4].

One fairly general choice for a covariance function which encodes belief in the smoothness of the underlying function, is

$$K(\mathbf{x}_i, \mathbf{x}_j) = \nu \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|_W^2\right) + \rho \quad (2)$$

where ρ is a scalar and the norm $\|\cdot\|_W$ is defined as $\|\mathbf{v}\|_W^2 = \mathbf{v}^T \mathbf{W} \mathbf{v}$ with \mathbf{W} a diagonal matrix having elements w_d . The hyperparameters w_d define an inverse length scale for the function in dimension d , ν gives the expected amplitude of the function, and ρ is a bias term that accommodates non-zero-mean functions. These (positive) hyperparameters control the way the function behaves between observed data points, and can be tuned using maximum likelihood or MAP methods.

3 Gaussian Process Value Iteration

As mentioned above, many authors have considered methods for finding approximately optimal policies for POMDPs by constructing approximate value functions which do not grow uncontrollably in complexity with time. A large subclass of such approximate methods involve computing the approximate value function at only a finite set of points in belief space and then performing value iteration on those (or another finite set of) points, extrapolating the value between known points where necessary. A few of these methods can be shown to converge or to provide bounds on the true value function, and some (notably curve-fitting methods which attempt to learn a parametrized form of the value function) are known to diverge on certain MDPs [2].

One major drawback to all of these methods is that in computing one step of value iteration (or “backup” operation) for a given point b in belief space, the values of all points reachable from b are treated equally in the sense that they are all presumed equally correct. For example, suppose we take our set of belief points at which we will try to learn the value function to be $\{b_1, b_2, b_3\}$. Suppose further that under the dynamics of our POMDP, b_2 and b_3 are absorbing belief states (they can only transition to themselves), and in one step b_1 can only reach (with equal probability) b_2, b_3 , and some fourth belief state b_4 . At every step of value iteration, the value at b_2 and b_3 will be exact, while the value at b_4 will be somehow interpolated from the others and may be wildly inaccurate. The value at b_1 is updated according to the values at b_2, b_3 , and b_4 weighted equally, with no accounting for the possibly great disparity in their accuracy.

It would be much better if there were a way to incorporate our knowledge about the accuracy of the value function at any point in belief space. This would enable the algorithm to avoid fitting uncertain points if doing so seems inconsistent with other information, and to be wary of areas in belief space promising large rewards if those areas are believed to be potentially inaccurate. Accounting for value function uncertainty in a Bayesian way requires that we introduce a prior distribution over the value function. This is the motivation behind the use of Gaussian processes to model the value function.

One might raise this objection to the use of Gaussian processes to model POMDP value functions: a Gaussian process prior can enforce neither convexity nor piecewise-linearity, and so we know from the start that we are choosing a prior which does not in fact capture our true beliefs about the function. However, with enough iterations, POMDP value functions often begin to look smooth, and in the infinite horizon they need not be piecewise linear at all. In fact, other authors have found a smooth approximation of the function, especially during the initial steps of approximate value iteration, to be advantageous[5]. A Gaussian process can also be given piecewise-linear trends through the appropriate choice of a covariance function. Furthermore, since we know the global length scale in belief space is about 1 in every dimension (belief states lie on the probability simplex), restricting the the inverse length scale hyperparameters of the covariance function to lie roughly between 0 and 1 will tend to cause the function to be either nearly convex or nearly concave in all dimensions, and observed data should enforce the former. Finally, given enough data points, Gaussian processes with suitably-chosen covariance functions can come very close to fitting most functions.

Another advantage of Gaussian processes is that by providing estimates of uncertainty it is possible to determine which areas of belief space deserve more exploration. Finally, as mentioned earlier it is straightforward to calculate the curvature of the mean function at any point. This can be used to diagnose belief states which locally violate the convexity constraint, again allowing further resources to be used to refine the value estimate.

Most of our proposed algorithm follows from the preceding discussion, but care must be taken to propagate uncertainty. We model each of the $|A|$ action-value functions $Q(\cdot, a)$ as a Gaussian process. At some iteration t , the action-value for any belief state is:

$$Q_t(b, a) = \sum_s b(s)R(s, a) + \gamma \sum_o P(o|a, b) \max_{\alpha} Q_{t-1}(\mathbf{SE}(b, a, o), \alpha) \quad (3)$$

What is the distribution of $Q_t(b, a)$? The $|\Omega||A|$ points $Q_{t-1}(\mathbf{SE}(b, a, o), \alpha)$ on which $Q_t(b, a)$ is based come from $|A|$ Gaussian processes, and each of those processes defines a multivariate normal distribution: $Q_{t-1}(\cdot, \alpha) \sim \mathcal{N}(\mu_{\alpha}, \Sigma_{\alpha})$ where we index the elements of μ_{α} and Σ_{α} with (bo) , e.g. $\mu_{\alpha, bo}$, bo being shorthand for $\mathbf{SE}(b, a, o)$. The major problem in computing the distribution over $Q_t(b, a)$ is the max operator. We consider two approximate ways for dealing with it.

In both methods we will assume the $|A|$ random vectors $Q_{t-1}(\cdot, \alpha)$ are independent of one another. In the first method, we approximate the max operator as simply passing through the random variable with the highest mean. That is,

$$\max_{\alpha} [Q_{t-1}(\mathbf{SE}(b, a, o), \alpha)] = Q_{t-1}(\mathbf{SE}(b, a, o), \alpha^*(bo))$$

where $\alpha^*(bo) = \operatorname{argmax}_{\alpha} \mu_{\alpha, bo}$. This approximation will be a good one when the means are more than a standard deviation or two from each other. In our algorithm, we will wish to compute the Q value $Q_t(\cdot, a)$ for N different

sample points. So the above approximation results in the joint distribution for $\mathbf{Q}_{t-1}^* \equiv [\max_{\alpha}[Q_{t-1}(b_1 o_1, \alpha)], \dots, \max_{\alpha}[Q_{t-1}(b_N o_{|\Omega|}, \alpha)]]^T$ (for an appropriate ordering) being Gaussian with mean and block-diagonal covariance matrix:

$$\mu_{\alpha^*} = \begin{bmatrix} \mu_{\alpha_1^*} \\ \vdots \\ \mu_{\alpha_{|A|}^*} \end{bmatrix}, \Sigma_{\alpha^*} = \begin{bmatrix} \Sigma_{\alpha_1^*} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Sigma_{\alpha_{|A|}^*} \end{bmatrix}$$

Here each of the $\mu_{\alpha_i^*}$ is a (possibly empty) vector containing all of the means $\mu_{\alpha_i, bo}$ for which $\alpha_i(bo) = \alpha^*(bo)$. Similarly, the $b_n o_j$ -th element of each (possibly empty) matrix $\Sigma_{\alpha_i^*}$ is taken from the matrix Σ_{α} for which $\alpha_i(b_n o_j) = \alpha^*(b_n o_j)$.

The second approximation method takes into account the effects of the max operator, but at the expense of ignoring correlations among the function values. We now consider only the marginals $Q_{t-1}(\mathbf{SE}(b, a, o), \alpha)$ with distributions $\mathcal{N}(\mu_{\alpha, bo}, \sigma_{\alpha, bo}^2)$ given by the Gaussian process for $Q_{t-1}(\cdot, \alpha)$. Clark [6] calculates the moments for the distribution of the max of two normally distributed variables q_1 and q_2 . When q_1 and q_2 are independent (as we've assumed for values from different Q-functions) with means and variances (μ_1, σ_1^2) and (μ_2, σ_2^2) , respectively, the first two moments of $\max(q_1, q_2)$ are given by:

$$\begin{aligned} E[\max(q_1, q_2)] &= \mu_1 \Phi(m) + \mu_2 \Phi(-m) + s\varphi(m) \\ E[(\max(q_1, q_2))^2] &= (\mu_1^2 + \sigma_1^2)\Phi(m) + (\mu_2^2 + \sigma_2^2)\Phi(-m) + (\mu_1 + \mu_2)s\varphi(m) \end{aligned}$$

where $s^2 = \sigma_1^2 + \sigma_2^2$, $m = (\mu_1 - \mu_2)/s$, and Φ is the cdf and φ is the pdf for a zero-mean, unit variance normal. Clark also gives evidence that this distribution may often be well-approximated by a Gaussian, suggesting a method for inductively computing the distribution of the max of three or more normal variables by taking max of the third with the max of the first two moment-matched to a normal, and so on. Applying this method to compute the distributions for all $N|\Omega|$ max operations yields a set of Q-values, each marginally distributed approximately as $\mathcal{N}(\mu_{\alpha^*(bo), bo}, \sigma_{\alpha^*(bo), bo}^2)$. If we now assume that all Q-values (even those from the same Q-function) are approximately independent, then \mathbf{Q}_{t-1}^* will again be distributed as a Gaussian with mean and diagonal covariance matrix:

$$\mu_{\alpha^*} = \begin{bmatrix} \mu_{\alpha^*(b_1 o_1), b_1 o_1} \\ \vdots \\ \mu_{\alpha^*(b_N o_{|\Omega|}), b_N o_{|\Omega|}} \end{bmatrix}, \Sigma_{\alpha^*} = \begin{bmatrix} \sigma_{\alpha^*(b_1 o_1), b_1 o_1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{\alpha^*(b_N o_{|\Omega|}), b_N o_{|\Omega|}}^2 \end{bmatrix}$$

Both methods produce a Gaussian approximation for the max of a set of normally distributed vectors. And since $\mathbf{Q}_t^a \equiv [Q_t(b_1, a), \dots, Q_t(b_N, a)]^T$ is related to \mathbf{Q}_{t-1}^* by a linear transformation via equation (3), we have that

$$\mathbf{Q}_t^a = \mathbf{g} + \mathbf{G}\mathbf{Q}_{t-1}^* \sim \mathcal{N}(\mathbf{G}\mu_{\alpha^*} + \mathbf{g}, \mathbf{G}\Sigma_{\alpha^*}\mathbf{G}^T)$$

where \mathbf{G} and \mathbf{g} represent the Bellman linear transform in equation (3).

So we have now arrived at the distribution for our Q-functions after one step of value iteration. These Q-value distributions form the training input for the Gaussian process at the next iteration, with the mean specifying the observed target values and the covariance matrix specifying the covariance of the Gaussian noise on those observations. Again, the clear advantage of the Gaussian process framework is the straightforward nature of doing regression with uncertain targets. In this way the uncertainty in the Q-function estimates can be propagated backwards through the Bellman backup operator in equation (3).

Now that we have explained our method for interpolating between sampled values, all that remains to discuss about the algorithm is the method for choosing the sample points: the belief points b_1, \dots, b_N at which the Q-values are computed at every iteration. There are many possible ways of heuristically choosing these points. Other authors have considered random selection of belief points, using the corners of the belief simplex and the most valuable points reachable from them in a few steps, adding promising mixtures of existing sample points, and various methods of stochastically simulating the belief MDP and using the points reached in these simulations. See [2],[7] for an overview.

In all problems we consider, an initial belief state with which the agent begins at time 0 is always specified. In this case, there may be infinitely many valid (in that they lie on the simplex) points in belief space which can never be reached, and we need only worry about “covering” in some sense the set of points reachable from the initial state. For this reason, we focused on stochastic simulation methods. Pineau, et al. [7] propose a method they call Stochastic Simulation with Exploratory Action which, for each belief point in the set, stochastically simulates every possible action and then adds only the point reached which is furthest (in the 1-norm sense) from the points already in the set. The idea is to prevent any belief point that our agent will encounter from being too far from the set, in which case the interpolation is likely to be inaccurate. This method does not make use of the agent’s current approximation to the value function at one stage to select the points for the next stage, and may focus on points which our agent will never encounter under its policy. Nevertheless, we found that it worked well as a method for choosing points. We also considered a hybrid of this method and simulation with ϵ -greedy action: after a few steps of value iteration under one set of belief points we would select a new set by simulating trajectories through belief space starting from the initial belief and using an ϵ -greedy policy with respect to our current Q-function approximations. The belief set would be created by starting with the initial point and repeatedly adding from the list of encountered belief points the point which was furthest (in the 1-norm) from the points already in the set, until some maximum number of points was reached.

| Method | tiger-grid | | hallway | | | hallway2 | | |
|-------------|------------|-------|---------|--------|-------|----------|--------|-------|
| | Reward | $ B $ | Goal% | Reward | $ B $ | Goal% | Reward | $ B $ |
| GP | 2.26 | 300 | 100 | 0.49 | 200 | 42 | 0.24 | 300 |
| Disabled GP | 1.78 | 300 | 71 | 0.38 | 300 | 37 | 0.24 | 300 |
| PBVI | 2.25 | 470 | 96 | 0.53 | 86 | 98 | 0.34 | 95 |
| PBUA | 2.30 | 660 | 100 | 0.53 | 300 | 100 | 0.35 | 1840 |

Table 1: Comparison of approximate value iteration methods on three standard POMDP problems. Figures given are the total discounted reward received, the number of belief points sampled, and the percentage of trials on which the goal state was reached (where applicable).

4 Empirical results

Following [7], we ran our algorithm on a series of three POMDP problems from [8] which have become somewhat standard for the testing of scalable POMDP algorithms. These are **hallway** (60 states), **hallway2** (92 states), and **tiger-maze** (36 states). We used the covariance function given by (2), plus an additional diagonal noise term to give the algorithm extra flexibility. For comparison, we also ran a handicapped version of our algorithm and implemented a version of the SPOVA algorithm proposed in [5]. The handicapped Gaussian process was not allowed to propagate its uncertainty, in order to explore the practical benefit of this ability. The SPOVA model assumes the softmax form $V(b) = [\sum_{\beta} (b \cdot \beta)^k]^{1/k}$ for the value function, for some value of k (smaller k corresponds to a smoother approximation) and set of hyperplanes $\{\beta\}$. In our version, the hyperplanes are learned at each step of value iteration by minimizing the squared error between the approximator and the value given by one backup over the set of observed belief points. This implementation demonstrates the pitfalls of curve-fitting algorithms which cannot use uncertainty information, as SPOVA rarely reached goal states and was left out of the table.. Table 1 summarizes the results. We have also included, as a reference, the results achieved by two other state-of-the-art algorithms (PBVI and PBUA) as reported in [7].

5 Conclusion and future work

We have presented an algorithm for approximate value iteration in POMDPs which uses Gaussian processes to propagate and interpret uncertainty in its approximations at each step. The algorithm was shown to perform competitively on standard problems which are considered fairly large and challenging, and certainly beyond the scope of exact methods at present. The exception was the **hallway2** problem, which had the largest state space. We believe our algorithm needed more than 300 belief point samples to perform at a competitive level, but computational constraints prevented that. The algorithm was also contrasted with another curve-fitting algorithm and with a version of itself

unable to propagate uncertainty in order to demonstrate the usefulness of this.

The method we presented can be extended in many ways. We mentioned earlier that the uncertainty measure and easy derivative calculations offered by Gaussian processes provide additional information for the selection of belief training points, but have not yet attempted to implement an algorithm which uses this information. The interpolation part of the algorithm can also be combined with many other point-selection procedures and can use a fixed set of sample belief points or be interleaved with belief set updates. Derivative “observations” can also be used to constrain the Gaussian process to have a given curvature or slope at the observed points, at the cost of having to solve larger linear systems.

We have so far only considered a curve-fitting type of approximate value iteration, but a vector-valued Gaussian process could be used to provide a distribution over hyperplanes so that the uncertainty propagation method could be extended to algorithms which make use of the piece-wise linearity of POMDP value functions to build up solutions in the form of sets of hyperplanes. Finally, the primary limitation of Gaussian processes is the need to invert an $N \times N$ matrix where N is the number of training points. If we want at least few sample points per dimension of belief space, this limits the application of the gaussian process method to POMDPs with only a few hundred states. However, there are many approximate inversion schemes (such as [9]) to be explored which might enable the method to handle thousands of states, such as the `tag` problem presented as a challenge in [7].

Acknowledgments

We would like to thank Michael Duff, Geoff Gordon, and Joelle Pineau for useful discussions about value iteration in POMDPs.

References

- [1] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
- [2] Milos Hauskrecht. Value function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.
- [3] David J.C. MacKay. Introduction to gaussian processes. Technical report, Cambridge University, 1997.
- [4] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen. Derivative observations in gaussian processes models of dynamic systems. In *NIPS 15*, 2003.
- [5] Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *IJCAI*, 1995.
- [6] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, March-April 1961.

- [7] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: And anytime algorithm for POMDPs. In *IJCAI*, 2003.
- [8] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: scaling up. In *ICML*, 1995.
- [9] J. Skilling. Bayesian numerical analysis. In Jr. W. T. Grandy and P. Milonni, editors, *Physics and Probability*. Cambridge University Press, Cambridge, 1993.