GP-POMDP: Bayesian Reinforcement Learning in Continuous POMDPs with Gaussian Processes

Patrick Dallaire, Stephane Ross and Brahim Chaib-draa

Abstract—Partially Observable Markov Decision Processes (POMDPs) provide a rich mathematical model to handle real-world sequential decision processes but require a known model to be solved by most approaches. However, mainstream POMDP research focuses on the discrete case and this complicates its application to most realistic problems that are naturally modeled using continuous state spaces. In this paper, we consider the problem of optimal control in continuous and partially observable environments when the parameters of the model are unknown. We advocate the use of Gaussian Process Dynamical Models (GPDMs) so that we can learn the model through experience with the environment. Our results on the blimp problem show that the approach can learn good models of the sensors and actuators in order to maximize long-term rewards.

I. INTRODUCTION

In the past few decades, Reinforcement Learning (RL) has emerged as an elegant and popular technique to handle decision problems when the model is unknown. Reinforcement learning is a general technique that allows an agent to learn the best way to behave, i.e. to maximize expected return, from repeated interactions in the environment. One of the most fundamental open questions in RL is that of exploration-exploitation: namely, how should the agent choose actions during the learning phase, in order to both maximize its knowledge of the model as needed to better achieve the objective later (to explore), and maximize current achievement of the objective based on what is already known about the domain (to exploit). Under some (reasonably general) conditions on the exploratory behavior, it has been shown that RL eventually learns the optimal action-selection behavior. However, these conditions do not specify how to optimally trade-off between exploration and exploitation such as to maximize utilities throughout the life of the agent, including during the learning phase, as well as beyond.

Model-based Bayesian RL (BRL) is a recent extension of RL that has gained significant interest from the AI community as it can jointly optimize performance during learning and beyond, within the standard Bayesian inference paradigm. In this framework, prior information about the problem (including uncertainty) is represented in parametric form, and Bayesian inference is used to incorporate any new information about the model. Thus, the exploration-exploitation problem can be handled as an explicit sequential decision problem, where the agent seeks to maximize future expected return with respect to its current uncertainty on the model. An important limitation of this approach is that making decision is significantly more complex since it involves reasoning about all possible models and courses of action. In addition, most work to date on this framework has been limited to cases where full knowledge of the agent’s state is available at every time step [1], [2], [3], [4].

Mainstream frameworks for model-based BRL for POMDPs [5], [6] are only applicable in domains described by finite sets of states, actions and observations, where the Dirichlet distribution is used as a posterior over discrete distributions. This is an important limitation in practice as many practical problems are naturally described by continuous states, actions and observations spaces. For instance, in robot navigation problems, the state is normally described by continuous variables such as the robot’s position, orientation, velocity and angular velocity; the choice of action controls the forward and angular acceleration, which are both continuous; and the observations provide a noisy estimate of the robot’s state based on its sensors, which would also be continuous.

One recent framework for model-based BRL in continuous POMDP is the Bayes-Adaptive Continuous POMDP [7], which extends their previously proposed Bayes-Adaptive POMDP model for discrete domains to continuous domains. However, this framework assumes that the parametric form of the transition and observation functions are known and that only the mean vector and covariance matrix of the noise random variables are unknown. Hence, this framework has limited applicability when the transition and observation functions are completely unknown.

This paper aims to investigate a model-based BRL framework that can handle domains that are both partially observable and continuous without assuming any parametric form for the transition, observation and reward function. To achieve that, we use Gaussian Process Dynamical Models (GPDMs) for model identification to learn these functions, and then propose a planning algorithm which selects the actions that maximizes long-term expected rewards under the current model.

II. RELATED WORK

In the case of continuous POMDPs, the literature is relatively sparse. In general, the common approach is to assume a discretization of the state space and therefore, be an incomplete model of the underlying system.

A first approach to continuous-state POMDPs is undoubtedly Thrun’s approach [8], where a belief is viewed as
a set of weighted samples which can be considered as a particular case (a degenerative case) of Gaussian mixture representation. The author presents a Monte Carlo algorithm for learning to act in POMDPs with real-valued state and action spaces, paying thus tribute to the fact that a large number of real-world problems are continuous in nature. A reinforcement learning algorithm value iteration is used to learn the value function over belief states. Then, a sample version of nearest neighbors is used to generalize across states.

A second approach to continuous-state POMDPs has been proposed by Porta et al. [9]. In this paper, the authors show that the optimal finite-horizon value function over the infinite POMDP belief space is piecewise linear and convex and is defined by a finite set of $\alpha$-functions, expressed as linear combinations of Gaussians, assuming the sensors, actions and rewards models are also Gaussian-based. Then, they show it for a fairly general class of POMDP models in which all functions of interest are modeled by Gaussian mixtures. All belief updates and value iteration backups can be carried out analytically and exactly. Experimental results, using the previously proposed randomized point-based value iteration algorithm PERSEUS [10], in a simple robot planning with a continuous domain, have been demonstrated. The same authors have extended their work to deal with continuous POMDPs as Porta et al. [9], assuming the POMDP belief space is piecewise linear and convex and is defined below, exists.

Another approaches related to our work turn around the modeling of time series data using dynamical systems [12]. In the case of nonlinear time series analysis, Wang and colleagues introduced Gaussian Process Dynamical Models (GPDMs) so that they can learn models of human pose and motion from high-dimensional motion capture data. Their GPDM is a latent variable model comprising a low dimensional latent space with associated dynamics, as well as a map from the latent space to an observation space. The authors marginalized out the model parameters in closed form by using Gaussian process priors for both the dynamical and observation mappings. It results in a nonparametric model for dynamical systems. Thus, their approach is sustained by a continuous hidden Markov model and does not include actions and rewards.

III. CONTINUOUS POMDP

We consider a continuous POMDP to be defined by the tuple $(S, A, Z, T, O, R, b_1, \gamma)$:

- $S \subset \mathbb{R}^m$: The state space, which is continuous and potentially multidimensional.
- $A \subset \mathbb{R}^n$: The action space, which is continuous and potentially multidimensional. It is assumed here that $A$ is a closed subset of $\mathbb{R}^n$, so that the optimal control, as defined below, exists.
- $Z \subset \mathbb{R}^p$: The observation space, which is continuous and potentially multidimensional.
- $T: S \times A \times S \rightarrow [0, \infty]$: The transition function which specifies the conditional probability density $T(s, a, s') = \Pr(s'|s, a)$ of moving to state $s'$, given the current state $s$ and the action $a$ performed by the agent.
- $O: S \times A \times Z \rightarrow [0, \infty]$: The observation function which specifies the conditional probability density $O(s', a, z') = \Pr(z'|s', a)$ of observing observation $z'$ when moving to state $s'$ after doing action $a$. Here we restrict our attention to the case where the conditional distribution $O(s', a, \cdot)$ is Gaussian for all state $s'$ and action $a$ and we denote $T^a_{s, s'}$ and $O^a_{s, z'}$ to be respectively the mean vector and covariance matrix of the distribution $O(s', a, \cdot)$.
- $R: S \times A \times \mathbb{R} \rightarrow [0, \infty]$: The reward function which specifies the conditional probability density $R(s', a, r') = \Pr(r'|s', a)$ of getting reward $r'$, given that the agent performed action $a$ and reached state $s'$. Here we restrict our attention to the case where the conditional distribution $R(s', a, \cdot)$ is Gaussian for all state $s'$ and action $a$ and we denote $R^a_{s, r'}$ and $T^a_{s, r'}$ to be respectively the mean vector and covariance matrix of the distribution $R(s', a, \cdot)$.
- $b_1 \in \Delta S$: The initial state distribution.
- $\gamma$: The discount factor.

The dynamical system is defined by the following equations:

$$
\begin{align*}
    s_1 & \sim b_1 \\
    s_t | s_{t-1}, a_{t-1} & \sim N(T^{a_{t-1}, a_1}_{s_t, s_{t-1}}, T^{s_{t-1}, a_{t-1}}_{s_t, s_{t-1}}) \\
    z_t | s_t, a_{t-1} & \sim N(O^a_{s_t, a_{t-1}}, O^a_{s_t, z_{t-1}}) \\
    r_t | s_t, a_{t-1} & \sim N(\gamma R^a_{s_t, r'}, R^a_{s_t, r'})
\end{align*}
$$

where $t$ is the time index, $s \in S$, $a \in A$, $z \in Z$ and $r \in \mathbb{R}$ is the reward.

The posterior distribution over the current state $s_t$ of the agent at any time $t$, denoted $b_t$ and referred to as the belief at time $t$, can be maintained via Bayes’ rule as follows:

$$
b_t(s') \propto O(s', a_{t-1}, z_t) \int S T(s, a_{t-1}, s') b_{t-1}(s) ds
$$

where $S$ is the set of states and $b_1(s')$ is the initial belief distribution.
The optimal policy is obtained by solving the Bellman equation:

\[
V^*(b) = \max_{a \in A} \left[ \int_S b(s) \int_S R^a(s, a, s') ds' ds + \gamma \int_Z f(z|b, a)V^*(b, z) dz \right]
\]  

where \( f(z|b, a) = \int_S O(s', a, z) \int_S T(s, a_{t-1}, s') b(s) ds' ds \) is the conditional probability density of observing \( z \) after doing action \( a \) in belief \( b \), and \( b^{a, z} \) denotes the next belief obtained by performing a Bayes' rule update of \( b \) with action \( a \) and observation \( z \).

IV. GP-POMDP

In this paper, we consider the problem of optimal control in such continuous POMDP where \( T, O \) and \( R \) are unknown. To consider making optimal decisions, the model is learned from sequence of action-observation through Gaussian Processes (GPs) modeling \([13],[14]\). GPs are a class of probabilistic models which focuses on points where a function is instantiated, through a Gaussian distribution over the function space. Usually a Gaussian distribution is parameterized by a mean and a covariance, and in the case of GPs, these two parameters are functions of the space on which the process operates.

For our problem of optimal control, we propose to use Gaussian Process Dynamical Model (GPDM) to learn the transition, observation and reward functions and then propose a planning algorithm which selects action that maximizes long-term expected rewards under the current model.

In order to use GPs to learn the transition, observation and reward model, we first assume that the dynamics can be expressed in the following form:

\[
\begin{align*}
    s_t &= T'(s_{t-1}, a_{t-1}) + \epsilon_T \\
    a_t &= O'(s_t, a_{t-1}) + \epsilon_O \\
    r_t &= R'(s_t, a_{t-1}) + \epsilon_R
\end{align*}
\]

(4)

where \( \epsilon_T, \epsilon_O \) and \( \epsilon_R \) are zero-mean Gaussian white noise. The functions \( T', O' \) and \( R' \) are respectively unknown deterministic function that returns the next state, observation and reward. We seek to learn this model and maintain a maximum likelihood estimate of the state trajectory by using a GPDM with optimization methods.

Let’s denote \( S = [s_1, s_2, \ldots , s_{N+1}]^T \) the state sequence matrix up to time \( t \), \( A = [a_1, a_2, \ldots , a_N]^T \) the action sequence matrix up to time \( t \), \( Z = [z_2, z_3, \ldots , z_{N+1}]^T \) the observation sequence matrix up to time \( t \) and \( r = [r_2, r_3, \ldots , r_{N+1}]^T \) the reward sequence vector up to time \( t \).

A. Gaussian Process Dynamical Model

The GPDM consist of a mapping from a latent space, which is assumed to follow first-order Markov dynamics, to an observation space. This probabilistic mapping, for the purpose of POMDPs is defined as \( S \times A \rightarrow Z \times R \), represent the observation-reward function where the actions are fully observable and the states are the latent variables. The dynamic mapping in latent space is \( S \times A \rightarrow S \) and correspond to transition function. Both mappings are defined as linear combinations of (nonlinear) basis functions:

\[
\begin{align*}
    s_t &= \sum_i b_i \phi(s_{t-1}, a_{t-1}) + n_s \\
    y_t &= \sum_j c_j \psi(s_t, a_{t-1}) + n_y
\end{align*}
\]

(5)

Here, \( B = [b_1, b_2, \ldots ] \) and \( C = [c_1, c_2, \ldots ] \) are weights for basis function \( \phi_i \) and \( \psi_j \), \( n_s \) and \( n_y \) are zero-mean time invariant white Gaussian noise. The joint observation-space is denoted as \( Y = Z \times R \) and therefore \( y_t \) is the observation vector \( z_t \) augmented with the reward \( r_t \). According to Bayesian methodology, the unknown parameters of the model should be marginalized out. This can be done in closed form \([15],[16]\) by specifying an isotropic Gaussian prior on the columns of \( C \) to yield a multivariate Gaussian likelihood

\[
p(Y | S, A, \alpha) = \int \frac{|W|^N}{\sqrt{(2\pi)^N(p+1)|K_Y|^{p+1}}} \exp \left( -\frac{1}{2} tr(K_Y^{-1}YW^2Y^T) \right)
\]

(6)

where \( Y = [y_2, y_3, \ldots , y_{N+1}]^T \) is a sequence of joint observations-rewards, \( K_Y \) is a kernel matrix computed with hyperparameters \( \alpha = \{\alpha_1, \alpha_2, \ldots , W\} \). The matrix \( W \) is diagonal and contain \( p + 1 \) scaling factor to account for different variances in observed data dimensions. Using a unique kernel function for both states and actions, the element of the kernel matrix are \( (K_Y)_{i,j} = k_Y([s_i, a_i], [s_j, a_j]) \).

Note that for the observation case, the action is the one done at the previous time step. The kernel used for this mapping is, with \( x = [s, a] \), the common radial basis function:

\[
k_Y(x, x') = \alpha_1 \exp \left( -\frac{\alpha_2}{2} \|x - x'\|^2 \right) + \alpha_3^{-1} \delta_{xx'}
\]

(7)

where hyperparameter \( \alpha_1 \) represent the output variance, \( \alpha_2 \) is the inverse with of the RBF which represent the smoothness of the function and \( \alpha_3 \) is the variance of the additive noise \( n_y \).

For the dynamic mapping on the latent space, similar methods are applied but needs to account for the Markov property. By specifying an isotropic Gaussian prior on the columns of \( B \), the marginalization can be done in closed form. The resulting probability density over latent trajectories is:

\[
p(S | A, \beta) = \frac{p(s_1)}{\sqrt{(2\pi)^N|K_X|^{n+N}}} \exp \left( -\frac{1}{2} tr(K_X^{-1}S_{out}S_{out}^T) \right)
\]

(8)

Here, \( p(s_1) \) is the initial state distribution \( b_1 \) and assumed isotropic Gaussian, \( S_{out} = [s_2, \ldots , s_{N+1}]^T \) is the matrix of latent coordinate which represent unobserved states. The \( N \times N \) kernel matrix \( K_X \) is constructed from \( X = [x_1, \ldots , x_N]^T \) where \( x_i = [s_i, a_i] \) with \( 1 \leq i \leq t - 1 \).
The kernel function for the dynamic mapping is:
\[
k_x(x, x') = \beta_1 \exp \left( -\frac{\beta_2}{2} ||x - x'||^2 \right) + \beta_3 x^T x' + \beta_4^{-1} \delta_{xx'}
\]  
(9)

where the additional hyperparameter \(\beta_3\) represents the output scale of the linear term. Since all hyperparameters are unknown and following [17], uninformative priors are applied so that \(p(\alpha) \propto \prod_i \alpha_i^{-1}\) and \(p(\beta) \propto \prod_i \beta_i^{-1}\). This result in a probabilistic interpretation of action-observation sequence:
\[
p(Z, r, S, \alpha, \beta|A) = p(Y|S, A, \alpha)p(S|A, \beta)p(\alpha)p(\beta)
\]  
(10)

To learn the GPDM, it has been proposed to minimize the joint negative log-posterior of the unknowns
\[
-\ln p(S, \alpha, \beta, W|Z, r, A) \text{ which is given by:}
\]
\[
L = \frac{m + n}{2} \ln |K_X| + \frac{1}{2} \text{tr}(K_X^{-1} S_{out} S_{out}^T) + \frac{1}{2} s_1^T s_1
\]
\[
- N \ln |W| + \frac{p + 1}{2} \ln |K_Y| + \frac{1}{2} \text{tr}(K_Y^{-1} Y W^T Y^T)
\]
\[
+ \sum_i \ln \alpha_i + \sum_i \ln \beta_i + C
\]  
(11)

For more details on GPDM learning methods, see [12]. The resulting MAP estimates of \(\{S, \alpha, \beta, W\}\) is then used to estimate the transition and observations-reward function:
\[
s_{t+1} = k_X([s_t, a_t])^T K_X^{-1} S_{out}
\]
\[
y_{t+1} = k_Y([s_{t+1}, a_t])^T K_Y^{-1} Y
\]  
(12)

Here \(k_X\) and \(k_Y\) denotes the vectors of covariance between test point and their respective training set w.r.t. to hyperparameters \(\alpha\) and \(\beta\) respectively. Since \(S\) is part of a MAP estimation, the states are assumed to be the most probable under the current model and therefore the last element is considered as the current state. This model estimate combined with the latent trajectory \(S\) correspond to our complete belief conditioned on observations, actions and rewards.

### B. Online Planning

By using the MAP estimate to update the belief of the agent conditioned on observations in the environment, we are able to address the question of how should the agent act given its current belief. To achieve this, we adopt an online planning algorithm which predicts future trajectories of actions, observations and rewards to estimate the expected return for each sampled immediate action. Each prediction is made with the corresponding Gaussian Process’ mean. Therefore, we approximate the value function with a finite horizon for each predicted states. Afterward, the agent simply performs the sampled immediate action which has highest estimate. The following algorithm describes the planning phase.

At time \(t\), the agent would call the function \(\text{PLANNING}(b_t, M, E, D)\), where \(b_t\) is its current belief, \(M\) is the number of actions to sample at the first level (selection) of the tree, \(E\) is the number of actions to sample for the next levels (evaluation) of the tree and \(d\) is the depth of the tree, i.e. the planning horizon. Then, the agent would execute the action \(a_t = A^*\) in the environment and then compute its belief \(b_{t+1}\) using function \(\text{LEARNGPDM}(b_t, a_t, r_{t+1})\) after observing \(z_{t+1}\) and \(r_{t+1}\). Then, the planning algorithm would be called again with the new belief \(b_{t+1}\) to choose the next action to perform, and so on.

### V. EXPERIMENTS

In this paper, we investigate the problem of learning to control the height of an autonomous blimp online, without knowing its pre-defined physical models. We have opted for blimps because, in comparison to other flying vehicles, they have the advantage of operating at relatively low speed, and they keep their altitude without necessarily moving. Furthermore, they are not sensitive to control errors as in the case of helicopters for instance [18]. In fact, the problem of controlling a blimp has been studied intensively in the past, particularly in the control community. Zhang and colleagues [19] have introduced a PID controller combined to a vision system to guide a blimp. For their part, Wyteh and Barron [20] have used a reactive controller; whereas Rao et al. [21] a controller using fuzzy logic. Some other researchers have used the non-linear dynamics to control several flight phases [22], [23].

All these approaches assume prior knowledge about the dynamics or pre-defined models. Our approach does not assume such prior knowledge and aims to learn the control policy directly, without passing through a priori information about the payload, the temperature, or the air pressure. In this context, the approach that is most closely to the one described here has recently been presented by Rottmann et al. [18]. Their approach applies Monte Carlo reinforcement learning and utilizes Gaussian processes for dealing with the continuous state-action space. More precisely, the approach assumes a completely observable continuous state-action space and utilizes the tabular Q-learning, two very restrictive assumptions in real applications. In fact, the blimp application is an illustrative example that sustains the applicability of the Bayesian Reinforcement Learning in Contin-

Algorithm 1 PLANNING(b, M, E, d)

1. if \(d = 0\) then Return 0, null
2. \(R^* \leftarrow -\infty\)
3. for \(i = 1\) to \(M\) do
4. Sample \(a \in A\) uniformly
5. Predict \(z|b, a\) and \(r|b, a\)
6. \(b' \leftarrow \text{LEARNGPDM}(b, a, z, r)\)
7. \(r', a' \leftarrow \text{PLANNING}(b', E, E, d - 1)\)
8. \(r \leftarrow r + r'\)
9. if \(r > R^*\) then \((R^*, A^*) \leftarrow (r, a)\)
10. end for
11. Return \((R^*, A^*)\)
uous POMDPs with Gaussian Processes; an approach that certainly can be used in any complex realistic application.

To evaluate the approach, we conducted experiments via simulation on the control of the height of an autonomous blimp [18]. The goal of our experiments is to validate the use of GPDMs for online model identification and state estimation combined with a planning algorithm for online decision-making. The agent aims to maintain a target height equals to 0 using as less energy as possible. Each episode starts at zero height and zero velocity and is run for 100 time steps. The time discretization is 1 second according to the simulator dynamics. Observations are the height (m) and the velocity (m/s) with an additive zero-mean Gaussian noise with 1 cm standard deviation. Rewards are corrupted with the same Gaussian noise. The blimp dynamics are simulated using a deterministic transition function where the resulting state, comprising height and velocity, is then degraded by a zero-mean isotropic Gaussian noise with 0.5 cm standard deviation. The continuous action set is defined as $A = [-1, 1]$ where the bounds represent maximum downward and upward actions.

To learn and plan from observations and rewards without prior knowledge, we trained a GPDM at each time step to provide the planning algorithm a model and associated believed trajectory. Since we are making several approximations during planning, we found that it was good to add some random exploration at early stages, to ensure we don’t converge toward a locally optimal model and act too greedily quickly. Hence, we execute a random-policy with probability $q = 0.9^t$, where $t$ is the current time step, or select best action found through planning with probability $1 - q$. The planning parameters were set to $M = 25$ samples for action selection level, $E = 10$ samples for state evaluation levels, $d = 3$ as tree depth and the discount factor $\gamma = 0.9$.

Figure 1 shows the averaged reward received by the agent. We notice that the curve stabilizes around a return of 0.8. This value corresponds to the received reward when a blimp is 5 cm apart the target height while not doing any significant action. In Figure 2, we observe that the blimp averaged distance from height 0 seems to stabilize around 10 cm.

The chosen reward function is set as the sum of two Gaussians. The first Gaussian having standard deviation of 1 meter gives half point for being roughly around the goal. The second one with 5 cm standard deviation gives another half point if the blimp is only a few centimeters apart its goal. A cost for actions is also incurred to force the blimp to reach its goal with minimal energy usage. The reward if finally corrupted by a Gaussian noise with 0.01 standard deviation.

Figure 3 shows the prediction error on the observations-reward sequence using equation (12) with the last state estimate. We defined the error here as the sum of absolute errors on the noisy observations and rewards. Figure 4 shows that most trajectories have large variance at the beginning and then the variance decrease as the agent gets more observations. A comparison to a random policy shows that this policy rapidly diverges to over 3 meters from the goal.

VI. DISCUSSION

The problem of optimal control in stochastic and partially observable environment with unknown model is a very important problem as a parametric model and its parameters are often difficult to specify. We have proposed the use of Gaussian Process Dynamical Models as a model identification tool in order to learn effective representation of the environment dynamics in continuous POMDPs. Although the agent has no prior information, our preliminary experimental results on the blimp problem show that the learned model, with the maximum likelihood estimates of the state trajectory, can provide useful predictions for our online planning algorithm. Most agents seem to achieve reasonable performance by receiving an averaged reward over 0.8 which we consider good under the chosen noise variances. The prediction error of 0.1 looks to confirm our intuition by noticing the fact that we sum the error of predicting 3 random variables having 0.01 standard deviation each.
The performance may have been better by using prediction uncertainty in the planning phase. This uncertainty could also be used for Bayesian exploration-exploitation trade-off. It would be interesting to adapt our approach in order to manage multiple sequences of actions-observations allowing the agent to learn from different episodes. The basic complexity of Gaussian processes with this increasing amount of data would require approximation methods to be computationally efficient. Moreover, with less time and data limitations, experiments could be conducted with reward functions farther of the optimal value function. Separate kernel functions for states and actions would also give better results, especially when the transition function is stochastic while actions are noise-free, as in our case.

REFERENCES


