An Expectation Maximization Algorithm for Continuous Markov Decision Processes with Arbitrary Rewards

Matt Hoffman, Nando de Freitas, Arnaud Doucet
{hoffmanm,nando,arnaud}@cs.ubc.ca
Univ. of British Columbia, Computer Science
Vancouver, B.C., Canada

Jan Peters
jan.peters@tuebingen.mpg.de
Max Planck Inst. for Biol. Cybernetics
Tübingen, Germany

Abstract

We derive a new expectation maximization algorithm for policy optimization in linear Gaussian Markov decision processes, where the reward function is parameterized in terms of a flexible mixture of Gaussians. This approach exploits both analytical tractability and numerical optimization. Consequently, on the one hand, it is more flexible and general than closed-form solutions, such as the widely used linear quadratic Gaussian (LQG) controllers. On the other hand, it is more accurate and faster than optimization methods that rely on approximation and simulation. Partial analytical solutions (though costly) eliminate the need for simulation and, hence, avoid approximation error. The experiments will show that for the same cost of computation, policy optimization methods that rely on analytical tractability have higher value than the ones that rely on simulation.

1 Introduction

A large variety of techniques have been proposed to attack the problem of optimal control in continuous action and state spaces. At one end of the spectrum, one encounters techniques that heavily exploit analytical tractability, such as linear quadratic Gaussian controllers (Bertsekas, 1995; Maciejowski, 2002). The restrictions of linear-Gaussian transition models and quadratic reward functions result in elegant and efficient recursions. However, these restrictions often prove to be too unrealistic in many practical domains.

At the other end of the spectrum, one finds techniques that rely on approximation and numerical computing. These include direct simulation and approximate dynamic programming; see for example (Baxter and Bartlett, 2001; Ng and Jordan, 2000; Thrun, 2000). The simulation approach is simple and very general, but can lead to poor results when the rewards are rare events or when there is a lack of a priori knowledge for constructing parameterized policies. Dynamic programming approaches, relying either on function approximation with stochastic approximation or naive discretization of the continuous states and actions, have not been found to perform well in high-dimensional settings despite recent developments and many years of research in the field of reinforcement learning. The strategy of mapping the control problem to one of statistical inference, as described in this paper, provides a fresh plan of attack on this hard problem.

Techniques in the middle of the spectrum, which take advantage of both analytical tractability and approximation methods, are very rare. One notable exception is the value iteration algorithm proposed in (Porta et al., 2006) for solving partially observed Markov decision processes (POMDPs) with a linear-Gaussian transition model and a mixture of Gaussians reward model. This specific representation enabled the authors to obtain closed-form alpha-function updates in the classical style of (Smallwood and Sondik, 1973). However, the approach requires that the action space be discretized. For many practical problems, where the action space is large, this discretization will suffer from the curse of dimensionality. If one adopts Monte Carlo discretization to avoid the curse, one is still subject to approximations errors: typically high variance. In this paper, we present an algorithm that uses a similar representation for the transition and reward models, but that does not require that the action space be discretized. The distributions over the action and state spaces will be obtained analytically.

Our approach follows from a recently proposed formulation of the stochastic planning and control problem as one of parameter estimation for suitable artificial statistical models (Toussaint and Storkey, 2006). There the authors propose a solution using the EM algorithm. This idea seems to have originated in (Dayan and Hinton, 1997), where only immediate rewards are

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5. Copyright 2009 by the authors.

considered. This formulation has since been applied to operational space control (Peters and Schaal, 2007) and in the sequential setting it has been studied by (Attias, 2003; Verma and Rao, 2006). Perhaps the most complete and clear formulation is the one of (Toussaint et al., 2006), which presents impressive results for finite state space models, although the authors only consider a single Gaussian reward function.

In sections 2 and 3 of this paper we will introduce the basic formulation of a sequential decision problem as one of statistical inference and outline an EM algorithm for estimating the policy parameters while analytically marginalizing over the states and actions. Section 4, is where the technical contribution of this paper begins: A new algorithm for arbitrary rewards that can be approximated with mixtures of Gaussians. After demonstrating how well the new algorithm performs on synthetic (but hard) MDPs, the paper concludes with a motivating robotics example that shows how to map a nonlinear control problem to a linear control problem with an arbitrary reward. Hence, if we can solve linear MDPs with arbitrary rewards, we can attack a large class of difficult nonlinear control problems.

2 Model specification

The goal is to perform policy optimization for discrete-time Markov decision processes (MDPs) defined by the $\mathcal{X}$-valued state-process $\{X_n\}_{n\geq 1}$ and $\mathcal{U}$-valued action-process $\{U_n\}_{n\geq 1}$. The initial-state, transition, and policy densities are given by:

$$\begin{gathered} X_1 \sim \mu(x_1), \\ X_{n+1} \,\big|\, (X_n = x_n, U_n = u_n) \sim f(x_{n+1}|x_n, u_n), \text{ and} \\ U_n \,\big|\, (X_n = x_n) \sim \pi_\theta(u_n|x_n) \end{gathered}$$

respectively. In order to ease notation later, we will also note that this model induces a Markov chain $\{Z_n\}_{n\geq 1}$ over the extended state-space $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$. We will refer to realizations of this chain as *state-action paths*, and denote their generative densities as

$$Z_1 \sim \mu_\theta(z_1) \quad \text{and} \quad Z_{n+1} \,\big|\, (Z_n = z_n) \sim f_\theta(z_{n+1}|z_n)$$

where $\mu_\theta(z_1) = \mu(x_1)\,\pi_\theta(u_1|x_1)$ and

$$f_\theta(z_{n+1}|z_n) = f(x_{n+1}|x_n, u_n)\,\pi_\theta(u_{n+1}|x_{n+1}).$$

Similarly, we can also define the short-hand notation $\pi_\theta(z_n) = \pi_\theta(u_n|x_n)$.

The policy π_θ as introduced earlier is assumed to have some parameterized form governed by θ. Given an *immediate reward* function $r(z) = r(x,u)$ the problem of solving an MDP then reduces to finding the values θ^* that maximize some measure of the total expected reward induced by this policy. For this work we will maximize the infinite-horizon, discounted, expected reward

$$J(\theta) = \mathbb{E}_{\mu_\theta(z_1)\, f_\theta(z_2|z_1)\cdots}\Big[\textstyle\sum_{k=1}^{\infty} \gamma^{k-1}\, r(Z_k)\Big]. \qquad (1)$$

The objective function $J(\theta)$ defined in (1) can also be reinterpreted as the objective function associated with an infinite mixture of finite horizon MDPs (where the reward only happens at the last time step). This was noted by (Toussaint and Storkey, 2006) and also used in (Doucet and Tadic, 2004) to solve integral equations. In order to discuss this interpretation we will first introduce the density

$$p_\theta(z_{1:k}|k) = \mu_\theta(z_1) \prod_{n=2}^{k} f_\theta(z_n|z_{n-1}), \qquad (2)$$

which represents the probability of a k-length state-action path. We can notice from Eq. (1) that use of the discount factor γ is very similar to putting a geometric distribution over path lengths k. Using this intuition we can write

$$p_\theta(k, z_{1:k}) = (1-\gamma)\gamma^{k-1}\, p_\theta(z_{1:k}|k) \qquad (3)$$

as the joint distribution over both paths and path lengths. We should note that this is a *trans-dimensional* distribution defined on $\biguplus_{k=1}^{\infty}\{k\} \times \mathcal{Z}^k$, i.e. a distribution where the dimensionality of the distribution (k in this case) is also a random variable. Given this formulation, the following proposition then holds:

Proposition 1. *The objective function $J(\theta)$ is proportional to the expected reward obtained at the last step under p_θ. More precisely,*

$$J(\theta) = (1-\gamma)^{-1}\, \mathbb{E}_{p_\theta}[r(Z_K)]. \qquad (4)$$

Proof. We can expand $J(\theta)$ from (1) as

$$\begin{aligned} J(\theta) &= \int \Big[\mu_\theta(z_1) \prod_{n=2}^{\infty} f_\theta(z_n|z_{n-1})\Big]\Big[\sum_{k=1}^{\infty} \gamma^{k-1}\, r(z_k)\Big]\, dz_{1:\infty}, \\ &= \sum_{k=1}^{\infty} \int (1-\gamma)^{-1}\,(1-\gamma)\,\gamma^{k-1}\, p_\theta(z_{1:k}|k)\, r(z_k)\, dz_{1:k} \\ &= (1-\gamma)^{-1}\, \mathbb{E}_{p_\theta}[r(Z_K)], \end{aligned}$$

from which our claim follows. □

Here the discount-factor γ induces a distribution to mix over finite time MDPS, where the stopping time is given by K. One benefit of this formulation is that the reward function is only "evaluated" at time K, i.e. we need only compute rewards that occur at the last time step. This representation will prove particularly useful later when performing inference as we will only need to evaluate the reward function at the end of the chain and then propagate this backwards in time.

3 Policy Search as Inference

The reformulated objective of Section 2 presents a new way of evaluating the expected reward which defines $J(\theta)$. In this section we will discuss an alternative method of *optimizing* this objective function, namely using methods originally developed for inference problems. We follow the approach of (Toussaint and Storkey, 2006) in this section and hence omit proofs. We will begin by constructing

$$\widetilde{p}(k, z_{1:k}|\theta) = \frac{p_\theta(k, z_{1:k})\, r(z_k)}{\mathbb{E}_{p_\theta}[r(Z_K)]}, \tag{5}$$

which we can easily see is the normalized density resulting from multiplying the reward $r(z)$ with the trans-dimensional distribution $p_\theta(k, z_{1:k})$. We should note that this formulation does require that $r(z)$ be strictly positive in order to ensure that the density is well defined[1]. This distribution was crucial to the development in (Hoffman et al., 2008) of a Markov Chain Monte Carlo (MCMC) procedure to sample from $\widetilde{p}(k, z_{1:k}, \theta) = p(\theta)\widetilde{p}(k, z_{1:k}|\theta)$, where $p(\theta)$ is a prior on the unknown parameters θ. By construction, this yields a marginal $\widetilde{p}(\theta) \propto J(\theta)\, p(\theta)$, resulting in higher probability mass where the expected reward is higher, and therefore samples which concentrate on these high reward areas. Instead of using this approximate integration method, in this paper we focus on developing a more efficient *exact* integration method for the specific mixture reward model presented in Section 4.

We will treat $(k, z_{1:k})$ as hidden data, and maximize the likelihood of θ via an Expectation Maximization (EM) algorithm. Using the standard EM terminology we can define the following terms:

- the *complete data likelihood* is given by $p_\theta(k, z_{1:k})\, r(z_k)$, i.e. the combined likelihood of our observed data (of which there is none) and hidden data;
- the *incomplete data likelihood* is the integral over our complete data likelihood with respect to our hidden data, and is thus given by $\mathbb{E}_{p_\theta}[r(Z_K)]$;
- the *predictive distribution* of our hidden data is the normalized distribution over hidden data, and thus $\widetilde{p}(k, z_{1:k}|\theta)$.

The EM algorithm is particularly useful in situations where it is difficult to directly optimize the incomplete data likelihood—often because of the existence of hidden data. Instead, we can iteratively maximize

$$Q(\theta, \theta_{i-1}) = \mathbb{E}_{\widetilde{p}}\big[\log\{p_\theta(k, z_{1:k})\, r(z_k)\}\big|\theta_{i-1}\big], \tag{6}$$

$$\theta_i = \arg\max_\theta Q(\theta, \theta_{i-1}). \tag{7}$$

Intuitively, at every iteration the previous values θ_{i-1} are used to calculate the expected complete data likelihood in the *E-step* (6) which is then maximized in the *M-step* (7). It is well known that this iterative technique is guaranteed to produce a local maximum of the incomplete data likelihood, which in our case is a maximum of $\mathbb{E}_{p_\theta}[r(Z_K)]$ as desired.

Since we will be maximizing the Q-function with respect to θ, we can drop any additive constants which don't depend on θ, simplifying the function to

$$Q(\theta, \theta_{i-1}) \eqsim \sum_{k=1}^{\infty} \widetilde{p}(k|\theta_{i-1}) \sum_{n=1}^{k} \int \widetilde{p}(z_n|k, \theta_{i-1}) \log \pi_\theta(z_n)\, dz_n$$

where '$\eqsim$' denotes 'equal up to an additive constant'. From this equation we can also see that the computational complexity of the E-step is $O(k_{\max}^2)$ per iteration for some maximum time-horizon $k_{\max}$. The hope, however, is that the analytic nature of these updates will allow for a fewer number of iterations. The following two sub-sections will describe the resulting E-step and the M-step in more detail.

3.1 The E-step

In order to construct the necessary E-step distributions we will utilize a technique similar to that used for parameter estimation in Hidden Markov Models (HMMs) and Linear Dynamical Systems (LDS). First, we will introduce the *forward messages* $\alpha_\theta(z_n)$ which will denote the distribution over state-action pairs after n steps. This can be defined recursively as

$$\alpha_\theta(z_n) = \int \alpha_\theta(z_{n-1})\; f_\theta(z_n|z_{n-1})\, dz_{n-1}, \tag{8}$$

where messages are initialized with the initial-state distribution, $\alpha_\theta(z_1) = \mu(z_1)$. Next, given some finite path length k, we can introduce the *backward messages* $\beta_\theta(z_n|k)$ which we will use to denote the expected reward in $n - k$ steps; here we can recursively define

$$\beta_\theta(z_n|k) = \int \beta_\theta(z_{n+1}|k)\; f_\theta(z_{n+1}|z_n)\, dz_{n+1} \tag{9}$$

where these messages are initialized with the immediate reward, $\beta_\theta(z_k|k) = r(z_k)$. In a standard HMM-context the backward messages would be used to represent the likelihood of all observed events from time $n + 1$ to k, but in our situation we have no observations other than the reward term. In essence we are treating the reward $r(z_k)$ as if it were the likelihood of some observed data that only happens at the end of our state-action path at time k.

Another useful property of these messages is that, unlike in an HMM-context, the α- and β-messages are independent of each other so long as the backward messages are parameterized using the form

$$\beta_\theta(z|\tau) = \beta_\theta(z_n|k) \text{ for } \tau = k - n. \tag{10}$$

In other words the β-messages denote the expected reward in τ steps, starting from state z. This was first observed by (Toussaint and Storkey, 2006), and allows us to compute these distributions in parallel. For the rest of this section we will continue using the notation given in (9) purely for reasons of exposition.

The forward and backward messages introduced above now provide us with an efficient means of calculating

[1] We will later show how to eliminate this assumption in some situations.

the expected reward for a given θ as well as the distributions required for the E-step.

Proposition 2. *The k-step reward is given by*

$$\mathbb{E}_{p_\theta}[r(Z_k)|k] = \int \alpha_\theta(z_n)\, \beta_\theta(z_n|k)\, dz_n \qquad (11)$$

for any n, and the infinite-horizon reward is given by

$$\mathbb{E}_{p_\theta}[r(Z_K)] = \sum_{k=1}^{\infty} (1-\gamma)\, \gamma^{k-1}\, \mathbb{E}_{p_\theta}[r(Z_k)|k]. \qquad (12)$$

Proposition 3. *The product of α- and β-messages gives us the unnormalized distribution over z_n,*

$$\widetilde{p}(z_n|k,\theta) = \frac{1}{\mathbb{E}_{p_\theta}[r(Z_k)|k]}\, \alpha_\theta(z_n)\, \beta_\theta(z_n|k), \qquad (13)$$

where the normalizing constant is the k-step reward. The distribution over k is given by

$$\widetilde{p}(k|\theta) = (1-\gamma)\gamma^{k-1}\, \frac{\mathbb{E}_{p_\theta}[r(Z_k)|k]}{\mathbb{E}_{p_\theta}[r(Z_K)]}. \qquad (14)$$

3.2 The M-step

In order to solve for θ_i at each iteration we must compute the gradient $\nabla Q(\theta, \theta_{i-1})$, which is equal to

$$\sum_{k=1}^{\infty} \widetilde{p}(k|\theta_{i-1}) \sum_{n=1}^{k} \int \widetilde{p}(z_n|k,\theta_{i-1}) \nabla \log \pi_\theta(z_n)\, dz_n. \qquad (15)$$

We can then analytically find the zeros of $\nabla Q(\theta, \theta_{i-1})$ if this function is concave with respect to θ. If an analytical solution is not possible a *generalized EM* (GEM) algorithm can be employed, where the gradient is evaluated at the current parameter values to obtain $\nabla Q(\theta_{i-1}, \theta_{i-1})$. Steps can then be taken in the direction of this gradient, a technique that is similarly guaranteed to converge to a local maximum (Lange, 1995). For this work we will use a quasi-Newton optimization approach, the LBFGS-B algorithm (Byrd et al., 1995).

It is also possible to make a direct connection between EM-based algorithms and the policy gradient, ∇J. By rearranging terms we can write the Q-function's gradient as

$$\begin{aligned}\nabla Q(\theta, \theta_{i-1}) &= \int \widetilde{p}(k, z_{1:k}|\theta_{i-1})\, \nabla \log p_\theta(k, z_{1:k})\, dz_{1:k}\, dk \\ &= \int \frac{p_{\theta_{i-1}}(k, z_{1:k})\, r(z_k)}{\mathbb{E}[r(Z_K)|\theta_{i-1}]} \cdot \frac{\nabla p_\theta(k, z_{1:k})}{p_\theta(k, z_{1:k})}\, dz_{1:k}\, dk,\end{aligned}$$

and by evaluating this gradient at θ_{i-1}, we obtain

$$\begin{aligned}\nabla Q &= \frac{1}{\mathbb{E}[r(Z_K)|\theta_{i-1}]} \int \nabla p_{\theta_{i-1}}(k, z_{1:k})\, r(z_k)\, dz_{1:k}\, dk \\ &= \frac{1}{\mathbb{E}[r(Z_K)|\theta_{i-1}]} \cdot (1-\gamma)\, \nabla J(\theta_{i-1})\end{aligned}$$

where the second line follows directly from Equation (4). This equivalence has several implications, the first of which is simply that a GEM algorithm computes a gradient in exactly the same direction as the policy gradient. Secondly, it shows how we can use the methods of this section to calculate the gradient even for reward-models that may not necessarily be positive, i.e. the gradient can be written as

$$\begin{aligned}\nabla J(\theta) &= \frac{\mathbb{E}[r(Z_K)|\theta]}{(1-\gamma)} \cdot \nabla Q(\theta, \theta) \\ &= \sum_k \gamma^{k-1} \sum_{n=1}^{k} \int \alpha_\theta(z_n)\, \beta_\theta(z_n|k) \nabla \log \pi_\theta(z_n)\, dz_n.\end{aligned}$$

This is well defined so long as the above integral exists. By performing this calculation we can obtain an analytical estimate of the gradient $\nabla J(\theta_{i-1})$, and hence we need not rely on simulation.

4 A mixture-of-Gaussians model

We will now describe a particular instance of the general EM algorithm described earlier. Consider state and action spaces given by $\mathcal{X} = \mathbb{R}^{n_x}$ and $\mathcal{U} = \mathbb{R}^{n_u}$. We will assume a linear-Gaussian[2] transition model and policy,

$$\begin{aligned}\mu(x_1) &= \mathcal{N}(x_1; \mu_0, \Sigma_0), \\ f(x_{n+1}|x_n, u_n) &= \mathcal{N}(x_{n+1}; Ax_n + Bu_n, \Sigma), \text{ and} \\ \pi_\theta(u_n|x_n) &= \mathcal{N}(u_n; Kx_n + m, \sigma^2 I).\end{aligned}$$

Here the policy is parameterized by $\theta = (K, m, \sigma)$, the model itself is paramaterized by $(\mu_0, \Sigma_0, A, B, \Sigma)$, and I is the identity matrix. We will further assume a reward model which is a combination of P unnormalized Gaussian functions

$$r(z) = \sum_{j=1}^{P} w_j\, \overline{\mathcal{N}}(y_j; M_j z, L_j), \quad z = [x; u] \qquad (16)$$

each parameterized by (w_j, y_j, M_j, L_j). It should be emphasized that these functions are only used for their functional form, and in particular each y_j is a parameter and should not be interpreted as a random variable. It is also worth noting that even were it normalized this is not *strictly* a Gaussian density in z because of the presence of M_j. Under this model the state-action transition model is also linear-Gaussian, given by

$$\begin{aligned}\mu_\theta(z_1) &= \mathcal{N}(z_1; \overline{\mu}_0, \overline{\Sigma}_0) \text{ and} \\ f_\theta(z_{n+1}|z_n) &= \mathcal{N}(z_{n+1}; \overline{F} z_n + \overline{m}, \overline{\Sigma}). \qquad (17)\end{aligned}$$

Although the parameters $(\overline{\mu}_0, \overline{\Sigma}_0, \overline{F}, \overline{m}, \overline{\Sigma})$ depend on θ, we have left this dependency implicit in order to simplify the notation. The exact form of the parameters is given in Figure 1, and we give these terms without proof as they are relatively simple to derive.

[2] We will let $\mathcal{N}(x; \mu, \Sigma)$ denote a Normal distribution in x with mean μ and covariance Σ, and let $\overline{\mathcal{N}}$ denote the unnormalized distribution.

State-action transition parameters:

$$\overline{F} = \begin{bmatrix} A & B \\ KA & KB \end{bmatrix},\ \overline{m} = \begin{bmatrix} 0 \\ m \end{bmatrix},\ \overline{\Sigma} = \begin{bmatrix} \Sigma & \Sigma K^{\mathrm{T}} \\ K\Sigma & K\Sigma K^{\mathrm{T}} + \sigma^2 I \end{bmatrix}$$

Initial state-action parameters:

$$\overline{\mu}_0 = \begin{bmatrix} \mu_0 \\ K\mu_0 \end{bmatrix},\ \overline{\Sigma}_0 = \begin{bmatrix} \Sigma_0 & \Sigma_0 K^{\mathrm{T}} \\ K\Sigma_0 & K\Sigma_0 K^{\mathrm{T}} + \Sigma \end{bmatrix}$$

Figure 1: Definition of the transition parameters.

Forward message recursion:

$$\begin{aligned} \widehat{\mu}_1 &= \overline{\mu}_0, & \widehat{\mu}_n &= \overline{F}\widehat{\mu}_{n-1} + \overline{m}, \\ \widehat{\Sigma}_1 &= \overline{\Sigma}_0; & \widehat{\Sigma}_n &= \overline{F}\widehat{\Sigma}_{n-1}\overline{F}^{\mathrm{T}} + \overline{\Sigma}. \end{aligned}$$

Backward message recursion:

$$\begin{aligned} \check{\Omega}_0^j &= M_j^{\mathrm{T}} L_j^{-1} M_j, \\ \check{\mu}_0^j &= M_j^{\mathrm{T}} L_j^{-1} y_j, \\ \check{c}_0^j &= \log|2\pi L_j| + y_j^{\mathrm{T}} L_j^{-1} y_j, \\ \check{\Omega}_{\tau+1}^j &= \overline{F}^{\mathrm{T}}(\overline{\Sigma}^{-1} - \overline{\Sigma}^{-1}\widetilde{\Sigma}\overline{\Sigma}^{-1})\overline{F}, \\ \check{\mu}_{\tau+1}^j &= \overline{F}^{\mathrm{T}}\overline{\Sigma}^{-1}(\widetilde{\Sigma}\overline{\Sigma}^{-1}\overline{m} + \widetilde{\Sigma}\check{\mu}_\tau^j - \overline{m}), \\ \check{c}_{\tau+1}^j &= \check{c}_\tau^j + \log|\overline{\Sigma}\widetilde{\Sigma}^{-1}| + \overline{m}^{\mathrm{T}}\overline{\Sigma}^{-1}\overline{m} \\ &\quad - (\check{\mu}_\tau^j + \overline{\Sigma}^{-1}\overline{m})^{\mathrm{T}}\,\widetilde{\Sigma}\,(\check{\mu}_\tau^j + \overline{\Sigma}^{-1}\overline{m}), \end{aligned}$$

where $\widetilde{\Sigma}^{-1} = \check{\Omega}_\tau^j + \overline{\Sigma}^{-1}$.

Figure 2: The α- and β-message recursions for the mixture of Gaussians model. In order to ease notation as much as possible we mark the statistics of the forward messages with a *hat* (e.g. $\widehat{a}$) and mark the backward pass statistics with a *check* (e.g. $\check{a}$).

With our model specified, we can now write the forward and backward messages for this problem:

$$\begin{aligned} \alpha_\theta(z_n) &= \mathcal{N}(z_n; \widehat{\mu}_n, \widehat{\Sigma}_n), \\ \beta_\theta(z|\tau) &= \sum_j w_j \exp\Big\{ -\tfrac{1}{2}(\check{c}_\tau^j + z^{\mathrm{T}}\check{\Omega}_\tau^j z - 2z^{\mathrm{T}}\check{\mu}_\tau^j)\Big\}. \end{aligned}$$

The full recursive definition can be seen in Figure 2. The updates for the forward message parameters are relatively simple, and are essentially the same as those given in the update phase of the discrete-time Kalman filter. The derivation of the backward messages is a more complicated (and tedious) process. Here we have reparameterized the individual Gaussian components of the reward model in *canonical form* and written the backward messages in the *time-to-go* notation.

For a given k and n, and $\tau = k - n$, the unnormalized distribution over z_n is given by the product of forward- and backward-messages

$$\alpha_\theta(z_n)\ \beta_\theta(z_n|\tau) = \sum_j w_j\, \widetilde{w}_{n\tau}^j\, \mathcal{N}(z_n; \widetilde{\mu}_{n\tau}^j, \widetilde{\Sigma}_{n\tau}^j), \quad (18)$$

where the parameters $(\widetilde{w}_{n\tau}^j, \widetilde{\mu}_{n\tau}^j, \widetilde{\Sigma}_{n\tau}^j)$ are defined in Figure 3. Further, via Proposition 3 we can obtain both the k-step reward and the predictive distribution,

$$\begin{aligned} \mathbb{E}_{p_\theta}[r(Z_k)|k] &= \int \alpha_\theta(z_n)\ \beta_\theta(z_n|\tau)\ dz_n = \sum_j w_j \widetilde{w}_{n\tau}^j \overset{\triangle}{=} \widetilde{w}_{n\tau}, \\ \widetilde{p}_\theta(z_n|k) &= \sum_j \nu_{n\tau}^j\, \mathcal{N}(z_n; \widetilde{\mu}_{n\tau}^j, \widetilde{\Sigma}_{n\tau}^j) \quad (19) \end{aligned}$$

where $\nu_{n\tau}^j = w_j \widetilde{w}_{n\tau}^j / \widetilde{w}_{n\tau}$. We can then write the mean and covariance of z_n given k as

$$\begin{aligned} \mathbb{E}[Z_n|k] &= \widetilde{\mu}_{n\tau} = \sum_j \nu_{n\tau}^j\, \widetilde{\mu}_{n\tau}^j, \text{ and} \\ \mathrm{cov}[Z_n|k] &= \widetilde{\Sigma}_{n\tau} = \sum_j (\nu_{n\tau}^j)^2\, \widetilde{\Sigma}_{n\tau}^j. \quad (20) \end{aligned}$$

Finally, by referring to Equation (14) we can easily obtain the discrete distribution $\widetilde{p}(k|\theta) \propto (1-\gamma)\,\gamma^{k-1}\,\widetilde{w}_{n\tau}$. Given $\theta_{i-1} = (K_{i-1}, m_{i-1}, \sigma_{i-1})$ we can now calculate the partial derivatives of $\log \pi_\theta(u|x)$ with respect to each policy parameter and plug these directly into Equation (15) to obtain $\nabla Q(\theta, \theta_{i-1})$, i.e.

$$\begin{aligned} \frac{\partial Q}{\partial K} &= \sum_k \widetilde{p}(k|\theta_{i-1}) \sum_{n=1}^k \sigma_{i-1}^{-2}\big(\mathbb{E}[U_n X_n^{\mathrm{T}}] - m_{i-1}\mathbb{E}[X_n^{\mathrm{T}}] \\ &\qquad - K_{i-1}\mathbb{E}[X_n X_n^{\mathrm{T}}]\big), \\ \frac{\partial Q}{\partial m} &= \sum_k \widetilde{p}(k|\theta_{i-1}) \sum_{n=1}^k \sigma_{i-1}^{-2}(\mathbb{E}[U_n] - K_{i-1}\mathbb{E}[X_n] - m_{i-1}), \\ \frac{\partial Q}{\partial \sigma} &= \sum_k \widetilde{p}(k|\theta_{i-1}) \sum_{n=1}^k \sigma_{i-1}^{-3}\mathbb{E}[C^{\mathrm{T}}C] - n_u \sigma_{i-1}^{-1}, \quad (21) \end{aligned}$$

where $C = U_n - K_{i-1}X_n - m_{i-1}$. We have also left the dependency on k in the above expectations implicit in order to shorten the notation. By setting this gradient equal to 0 and solving for θ we can obtain the EM update. Given the fact that $Z_n = [X_n; U_n]$, the expectations needed to perform these calculations can be trivially obtained from the sufficient statistics in (20), where

$$\begin{aligned} \mathbb{E}[Z_n Z_n^{\mathrm{T}}] &= \widetilde{\Sigma}_{n\tau} + \widetilde{\mu}_{n\tau}(\widetilde{\mu}_{n\tau})^{\mathrm{T}} \text{ and} \\ \mathbb{E}[C^{\mathrm{T}}C] &= \mathrm{Tr}(K_{i-1}\mathrm{cov}[X_n]K_{i-1}^{\mathrm{T}} + \mathrm{cov}[U_n]) \\ &\qquad + \|\mathbb{E}[U_n] - K_{i-1}\mathbb{E}[X_n] - m_{i-1}\|^2. \end{aligned}$$

The gradient ∇J can also be calculated, where

$$\begin{aligned} \nabla J(\theta_{i-1}) = \sum_k \gamma^{k-1} \sum_{n=1}^k \sum_j (w_j \widetilde{w}_{n\tau}^j)\,\cdot \\ \int \mathcal{N}(z_n; \widetilde{\mu}_{n\tau}^j, \widetilde{\Sigma}_{n\tau}^j)\nabla \log \pi_{\theta_{i-1}}(z_n)\ dz_n. \end{aligned}$$

The integral in this formulation can be evaluated using sufficient statistics similar to those in (20). And here we have made no assumptions as to the sign of w_j.

Parameterization of the predictive distr.:

$$
\begin{aligned}
\widetilde{\Sigma}^j_{n\tau} &= (\widehat{\Sigma}_n^{-1} + \check{\Omega}^j_\tau)^{-1} \\
\widetilde{\mu}^j_{n\tau} &= \widetilde{\Sigma}^j_{n\tau}(\widehat{\Sigma}_n^{-1}\widehat{\mu}_n + \check{\mu}^j_\tau) \\
\widetilde{w}^j_{n\tau} &= |\widehat{\Sigma}_n^{-1}\widetilde{\Sigma}^j_{n\tau}|^{\frac{1}{2}} \exp\Big\{ -\tfrac{1}{2}\Big[\check{c}^j_\tau + \widehat{\mu}_n^{\mathrm{T}}\widehat{\Sigma}_n^{-1}\widehat{\mu}_n \\
&\qquad\qquad - (\widetilde{\mu}^j_{n\tau})^{\mathrm{T}}(\widetilde{\Sigma}^j_{n\tau})^{-1}(\widetilde{\mu}^j_{n\tau})\Big]\Big\}
\end{aligned}
$$

Figure 3: Parameterization of $\widetilde{p}(z_n|k,\theta)$ as in (19).

5 Results on synthetic data

In this section we will empirically observe the behavior of EM on Gaussian MDPs of the form introduced in Section 4. The simplest way to test these methods involves randomly generating the parameters of an MDP model and observing their convergence behavior. We generated transition parameters A_{ij} and B_{ij} from a standard uniform and components of the initial-state mean μ_0 uniformly in the range $[0, 5]$; covariance terms for these models were initialized diagonally with standard-deviations uniformly sampled in the range $(0, 5]$. The reward terms w_j and y_j were initialized similarly, M_j was the identity, and the "covariances" L_j were initialized using a random SPD matrix with eigenvectors uniformly distributed in $(0, 5]$.

As an aside, it is also worth noting that although we know the optimal policy will be deterministic, by allowing the exploration term σ to vary we obtain annealing-like behavior where local maxima are smoothed out by a large initial value of σ. The plot in Figure 4 contains the average convergence behavior of the two EM variants on a series of 100 simple, 1-dimensional MDPs (a 3-dimensional parameter space). The trace of each optimization process is normalized to be in the range $[0, 1]$ and averaged across all models. The first thing to note is the poor performance of the standard EM algorithm as compared to the GEM algorithm. This behavior results from the high proportion of hidden data and is a situation that only worsens as the dimensionality increases.

We also contrast the EM-based approach to policy gradient methods including: (i) a gradient-free approach using *finite-differences* and common random numbers for variance reduction (i.e. PEGASUS (Ng and Jordan, 2000)); (ii) stochastic gradient ascent using the *vanilla policy-gradient* and the vanilla policy-gradient combined with the *optimal baseline*; (iii) the *natural actor-critic* (Peters and Schaal, 2008). The convergence rates of these different algorithms can also be seen in Figure 4. Given a well chosen learning rate—and if the reward model induces a nice, broad surface with well-defined gradients—the policy-gradient methods perform quite well. It is worth noting, however, that these algorithms are greatly affected by the choice of learning rate. For most models it seemed possible to vary the learning rate of the policy-gradient methods in order to achieve performance comparable with GEM, but these learning rates did not generalize across multiple models and required multiple runs to obtain. One other tradeoff, however, is the time-complexity of these two classes of algorithms. Each iteration of the policy gradient algorithms runs in time $O(pk_{\max})$ where p is the number of trajectories, and $k_{\max}$ is the time-horizon; the EM-based algorithms are $O(k^2_{\max})$. As a result, even accounting for the issue of learning rates, the fact that these algorithms are linear in $k_{\max}$ may make them seem more attractive for problems with a large time-horizon.

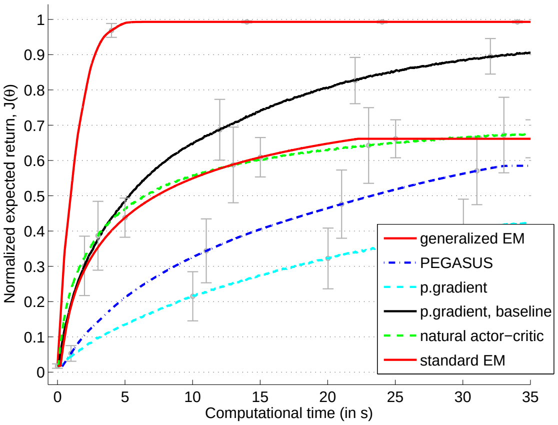

Figure 4: Convergence results on 100, randomly sampled 1-dimensional MDPs (for which the policy space is $\mathbb{R}^3$). For each algorithm we use the same tuning parameters (i.e. learning rate, number of trajectories, etc.) across different models. Also shown is the variation of this performance between different models.

The differences in performance, however, are much greater when the reward function is rare, i.e. with support limited to only a small region of the state space. Figure 5 shows the convergence behavior of the algorithms on a set of larger, 3-dimensional state- and action-space models (policies in these models are parameterized by $\theta \in \mathbb{R}^{13}$). In particular we see that the policy-gradient algorithms perform very poorly, and are for the most part unable to make any progress towards the goal. This is because in this space the reward model is much more rare than in lower dimensions, resulting in little-to-no gradient information except in a small region around the optimum and what little gradient exists is also likely washed out by the noise in the system. The GEM algorithm does not suffer from this problem not only because the iterations are analytic, but also because of the backward messages. In contrast, the policy gradient algorithms only perform a noisy forward pass, and so are much less likely to get good information about a rare reward. We also experimented with starting the initial value of θ closer to the optimum, as can be seen from the top plots of Figure 5, and we see that the the policy

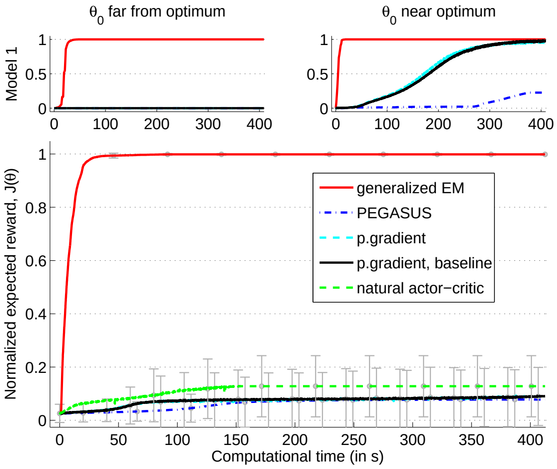

Figure 5: Convergence for 20 randomly selected MDPs (3D states and 3D actions, with policy space being $\mathbb{R}^{13}$). The policy gradient algorithms did not have enough gradient information to make any progress in all but a few of the models. The two plots on top show the change in this behavior when the initial policy θ_0 is initialized closer to the optimum.

gradients are able to make progress in this situation. This is not, however, an effective strategy: in order to find a closer value for θ_0 we were forced to initially solve the system using the GEM algorithm.

6 Robotic applications

Consider an n-jointed robotic system such that $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^n$ denote the joint angles, velocities, and accelerations respectively. Most such systems can be described by the rigid-body dynamics

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau} - \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{g}(\mathbf{q})), \tag{22}$$

where $\mathbf{M}(\mathbf{q})$ denotes the inertia matrix, $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ denotes the coriolis and centripetal forces, $\mathbf{g}(\mathbf{q})$ is the force due to gravity, and $\boldsymbol{\tau}$ the torques generated by the motors. The objective is then to control the evolution of the joints $[\mathbf{q}; \dot{\mathbf{q}}; \ddot{\mathbf{q}}] \in \mathbb{R}^{3n}$ with actions given by some sequence of torques $\boldsymbol{\tau} \in \mathbb{R}^n$. It is easy to see from (22), however, that the dynamics of this system are highly non-linear and as a result we cannot apply the techniques of Section 4. But the system can instead be reformulated in a different action-space that enforces linear dynamics.

A system can be called *feedback-linearizable* if there exists some function $\hat{\boldsymbol{\tau}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ that cancels the natural dynamics of the system in some local neighborhood of $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, where $\ddot{\mathbf{q}}$ is some desired joint-space acceleration. That is, we want a function $\hat{\boldsymbol{\tau}}$ that locally approximates the torque required to maintain some acceleration $\ddot{\mathbf{q}}$ from the state $(\mathbf{q}, \dot{\mathbf{q}})$. In general this function can be obtained via an estimate of the inverse dynamics (Lewis et al., 2004). Given the inverse dynamics, we can control the evolution of states $x = [\mathbf{q}; \dot{\mathbf{q}}]$ via actions $u = \ddot{\mathbf{q}}$ where the dynamics are linear.

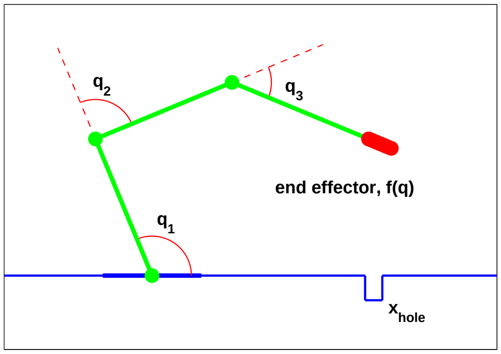

Figure 6: Model of a robot arm “peg-in-hole” task.

While it is often possible to linearize the dynamics of such a system, this can drastically change the reward model necessary to induce some desired behavior. It is also frequently the case that the reward model depends on non-linear terms such as the end-effector position $\mathbf{z} = \mathbf{f}(\mathbf{q})$. The function $\mathbf{f}$ denotes the forward-kinematics of the system, and depends on parameters such as the link lengths $\mathbf{l}$ and masses $\mathbf{m}$ (Lewis et al., 2004). Again, although the reward may be a simple quadratic in $\mathbf{z}$, this will in general be non-linear in x and u. These are precisely the situations where the mixture-of-Gaussians approach presented earlier should prove useful. On one hand this formulation immediately allows us to tackle multimodal regulation tasks whose rewards can be specified in the form of Equation (16). Another possibility, though, would be to fit the model either to some known functional form or to data.

One simple application of this technique on a robotic model is the “peg-in-hole” task, a depiction of which is shown in Figure 6. The goal of this task is to move the end-effector into some position $\mathbf{x}_{\text{hole}}$ and regulate about this point. We can specify the reward as

$$r(x, u) = \exp\left\{-\lambda_1 \|\mathbf{f}(\mathbf{q}) - \mathbf{x}_{\text{hole}}\|^2 - \lambda_2 \|\dot{\mathbf{q}}\|^2 - \lambda_3 \|\ddot{\mathbf{q}}\|^2\right\}.$$

Rather than attempting to fit this entire model, we can instead restrict ourselves to the reward-terms depend on $\mathbf{q}$. In this paper we fit the model using sparse, psuedo-input Gaussian Processes regression (Snelson and Ghahramani, 2006). The regression process results in a linear-combination of Gaussians with parameters (w_j, m_j, S_j) that can be used to approximate the full reward model:

$$\begin{aligned} r(x, u) &\approx \sum_j w_j\, \overline{\mathcal{N}}(\mathbf{q}; m_j, S_j) \cdot \exp\left\{-\lambda_2 \|\dot{\mathbf{q}}\|^2 - \lambda_3 \|\ddot{\mathbf{q}}\|^2\right\} \\ &= \sum_j w_j\, \overline{\mathcal{N}}([x; u]; y_j, L_j), \end{aligned}$$

for $y_j = [m_j; \mathbf{0}; \mathbf{0}]$ and $L_j = \text{blkdiag}(S_j, \frac{1}{\lambda_2} I, \frac{1}{\lambda_3} I)$. Given this information we can now perform policy search over the 22-dimensional space of all policies. Here we use a maximum time-horizon $k_{\max} = 100$.

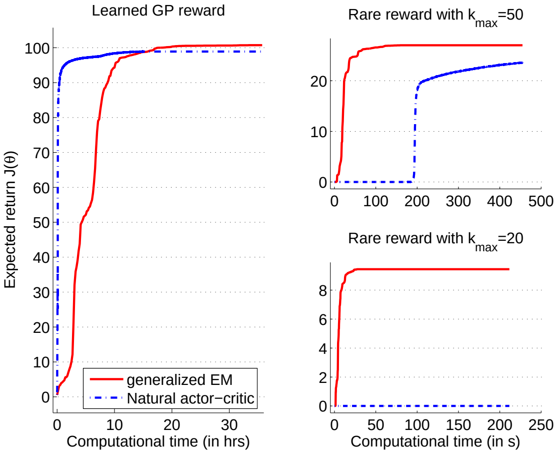

Figure 7: A trace of the optimization process for the 3-jointed robot arm model using the learned reward (left) and a rare reward (right).

Figure 7 shows a trace of the resulting optimization process; the resulting policy is successfully able to move the arm's end-effector to the position $\mathbf{x}_{\text{hole}}$ and from there regulates about this point. Here, although not a simple problem, the reward is relatively broad, and the natural actor-critic was able to do well. Due to the analytic nature of the GEM approach, however, we are still able to improve on this policy in the long-run. The GEM algorithm actually converges quite quickly, but each iteration is adversely affected by the $O(k_{\text{max}}^2)$ time complexity. Based on these results we also experimented with putting a single, very peaked reward at the resulting target joint angles. These results are also shown in Figure 7, and we can see that the analytic GEM algorithm greatly outperforms the natural actor-critic with rare rewards.

7 Conclusion

In this paper we derived an analytic approach to control problems whose rewards are modeled by a linear combination of Gaussians. We have also shown that when the reward is rare, a backwards pass is crucial to solving these types of problems.

References

H. Attias. Planning by probabilistic inference. In *UAI*, 2003.

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *JAIR*, 15:319–350, 2001.

D. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 1995.

P. Dayan and G. Hinton. Using EM for reinforcement learning. *Neural Computation*, 9:271–278, 1997.

A. Doucet and V. Tadic. On solving integral equations using Markov Chain Monte Carlo methods. Technical Report CUED-F-INFENG 444, Cambridge University Engineering Department, 2004.

M. Hoffman, A. Doucet, N. de Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *NIPS*, 2008.

M. Klaas, M. Briers, N. de Freitas, A. Doucet, and S. Maskell. Fast particle smoothing: If I had a million particles. In *ICML*, 2006.

K. Lange. A quasi-Newton acceleration of the EM algorithm. *Statistica Sinica*, 5(1):1–18, 1995.

F. Lewis, D. Dawson, and C. Abdallah. *Robot Manipulator Control: Theory and Practice*. CRC Press, 2004.

J. Maciejowski. *Predictive control with constraints*. Prentice-Hall, 2002.

A. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *UAI*, 2000.

J. Peters and S. Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

J. Peters and S. Schaal. Reinforcement learning for operational space control. In *ICRA*, 2007.

M. Porta, N. Vlassis, M. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *JMLR*, 7:2329–2367, 2006.

R. Smallwood and E. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 1973.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, 2006.

S. Thrun. Monte Carlo POMDPs. In *NIPS*, 2000.

M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *ICML*, 2006.

M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, University of Edinburgh, School of Informatics, 2006.

D. Verma and R. P. N. Rao. Planning and acting in uncertain environments using probabilistic inference. In *IROS*, 2006.