Learning Model-free Robot Control by a Monte Carlo EM Algorithm

Nikos Vlassis[1] Marc Toussaint[2] Georgios Kontes[1]
Savas Piperidis[1]

[1]Technical University of Crete
Dept. of Production Engineering and Management
73100 Chania, Greece
Email: {vlassis,kontes,savas}@dpem.tuc.gr

[2]TU Berlin, Franklinstr 28/29 FR6-9, 10587 Berlin, Germany
Email: mtoussai@cs.tu-berlin.de

Published in Autonomous Robots 27(2):123-130, 2009.

Abstract

We address the problem of learning robot control by model-free reinforcement learning (RL). We adopt the probabilistic model of Vlassis and Toussaint (2009) for model-free RL, and we propose a Monte Carlo EM algorithm (MCEM) for control learning that searches directly in the space of controller parameters using information obtained from randomly generated robot trajectories. MCEM is related to, and generalizes, the PoWER algorithm of Kober and Peters (2009). In the finite-horizon case MCEM reduces precisely to PoWER, but MCEM can also handle the discounted infinite-horizon case. An interesting result is that the infinite-horizon case can be viewed as a 'randomized' version of the finite-horizon case, in the sense that the length of each sampled trajectory is a random draw from an appropriately constructed geometric distribution. We provide some preliminary experiments demonstrating the effects of fixed (PoWER) vs randomized (MCEM) horizon length in two simulated and one real robot control tasks.

1 Introduction

Reinforcement Learning (RL) is a notable paradigm for robot control with several reported successes in recent years (Tedrake et al., 2005; Abbeel et al., 2007; Kober and Peters, 2009; Riedmiller et al., 2009; Martinez-Cantin et al., 2009). In this paper we address the problem of model-free RL, in which the goal is to learn a controller without knowledge of the dynamics of the system (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). In model-free RL, a controller is chosen from a parametric family and its parameters are learned by trial and error: for given controller parameters, the robot executes a number of trajectories through the state space, then uses the collected information (observed states, actions, and rewards) to update the controller parameters, and so forth. The challenge here is to estimate a good parameter update from a small number of sampled trajectories.

Several successful model-free RL approaches in robotics are instances of actor-critic or policy gradient algorithms (Ng and Jordan, 2000; Tedrake et al., 2005; Peters and Schaal, 2008a). A policy gradient RL algorithm computes and follows in each step the stochastic gradient of the policy performance until convergence at a local maximum. Peters and Schaal (2008b) describe several policy gradient algorithms that can be used in robotics. Such algorithms have been analyzed extensively and they have been very successful in practice, however their main shortcoming is that they depend on a learning rate which is difficult to set.

Dayan and Hinton (1997) suggested that a RL problem can be tackled by the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The main idea is to treat immediate rewards as probabilities of some fictitious events (an idea suggested earlier by Cooper (1988)) in which case one can use probabilistic inference techniques like EM for optimization. Recently, Kober and Peters (2009) developed an EM-based algorithm, called PoWER, for learning parametrized robot control policies in an episodic RL setting. PoWER inherits many of the advantages of the EM algorithm like simplicity of implementation and no need for a learning rate, and in several robotic tasks it demonstrated convincingly better performance than several state-of-the-art policy gradient algorithms.

PoWER is a promising new approach in learning robot control, but it is limited to episodic tasks that involve a fixed horizon of control actions. In this paper we draw on the probabilistic view of model-free RL (Toussaint and Storkey, 2006; Vlassis and Toussaint, 2009) and we re-derive the PoWER algorithm as an instance of a Monte-Carlo EM algorithm (MCEM) for inference on an infinite mixture model. The probabilistic view of model-free

RL allows for a generalization of the PoWER algorithm to the discounted infinite-horizon setting, which opens up the way for the use of EM-based techniques for learning time-invariant controllers. An interesting result of our framework is that the infinite-horizon case can be viewed as a 'randomized' version of the finite-horizon case, in the sense that the length of each sampled trajectory is a random draw from an appropriately constructed geometric distribution over the time variable. Simply put, PoWER can be turned into an infinite horizon algorithm just by randomization over the length of each trajectory (with an additional weighting of the rewards). We provide some preliminary experiments demonstrating the effects of fixed (PoWER) vs randomized (MCEM) horizon length in two simulated and one real robot control tasks.

2 Model-free RL as probabilistic inference

We model the robotic task as an infinite-horizon discrete-time Markov Decision Process (MDP) with continuous states $x \in \mathbb{R}^n$ and actions $u \in \mathbb{R}$ (the generalization to high-dimensional actions is straightforward but we limit our discussion to one-dimensional actions for clarity). The robot always starts from a state x_0 drawn from a distribution $p(x_0)$, and it follows a stochastic policy $\pi_\theta(u|x)$ parametrized by parameters θ. At each time step t the robot collects immediate reward r_t that is a function of the state x_t and the action u_t. In this work we are interested in the model-free setting in which we do not have access to the transition model of the MDP but we can sample trajectories from the MDP starting from $p(x_0)$ and following some policy. The discounted infinite horizon RL problem is defined as follows: Using only sampled experience from the MDP, find a point estimate of θ that maximizes the expected future reward (value of policy)

$$J(\theta) = E\Big[\sum_{t=0}^{\infty} \gamma^t r_t; \theta\Big], \tag{1}$$

where the expectation is taken over all possible trajectories through the MDP starting from $p(x_0)$ and following π_θ. In this paper we are only interested in model-free RL algorithms that do not attempt to build any models during execution, neither of the dynamics of the MDP nor of any auxiliary state or state-action value function.

When a model of the MDP is available, and the rewards r_t are nonnegative quantities (say, after normalization holds $r_t \in [0, 1]$), Toussaint and Storkey (2006) showed that it it possible to cast the optimization of $J(\theta)$ as

a problem of probabilistic inference on an infinite mixture of finite-horizon MDPs. In this approach, the horizon of the MDP is viewed as a discrete random variable T with geometric prior distribution $p(T) = (1-\gamma)\gamma^T$, for $T = 0, 1, \ldots, \infty$. The key idea, which goes back to Cooper (1988), is to treat rewards as probabilities of some fictitious events. In particular, the reward r_T obtained at some step T of the infinite-horizon MDP is viewed as the probability of some event R occurring at the last time step of a T-horizon MDP, where the two MDPs have identical dynamics. Let $\xi = (x_0, u_0, .., x_T, u_T)$ be a state-action trajectory of length $|\xi| = T$, and $\mathcal{X}^T = \{\xi : |\xi| = T\}$ be the space of length-T trajectories. In this case the expected return $J(\theta)$ becomes a likelihood function of an infinite mixture model:

$$J(\theta) = \sum_{t=0}^{\infty} \gamma^t E[r_t; \theta] = \frac{1}{1-\gamma} \sum_{t=0}^{\infty} p(t) \int_{\xi \in \mathcal{X}^t} p(\xi|t;\theta)\ p(R|\xi), \tag{2}$$

where $p(t) = (1-\gamma)\gamma^t$, for $t = 0, 1, \ldots, \infty$, is a geometric distribution, $p(\xi|t;\theta)$ the distribution of t-length trajectories $\xi \in \mathcal{X}^t$ (parametrized by policy parameters θ), and $p(R|\xi) = r_{|\xi|}$ is the probability of R occurring at the last time step of trajectory ξ (which is the terminal reward $r_{|\xi|}$ of ξ). Under the MDP dynamics, the distribution of T-length trajectories under policy π_θ is given by

$$p(\xi|T;\theta) = p(x_{\xi 0})\ \pi_\theta(u_{\xi T}|x_{\xi T}) \prod_{t=0}^{T-1} p(x_{\xi(t+1)}|x_{\xi t}, u_{\xi t})\ \pi_\theta(u_{\xi t}|x_{\xi t}), \tag{3}$$

where $p(x_{t+1}|x_t, u_t)$ is the transition model of the MDP, and $x_{\xi t}, u_{\xi t}$ are the observed states and actions in the trajectory ξ.

When the model of the MDP is known, maximization of J over θ can be carried out, for instance, by the EM algorithm (Toussaint and Storkey, 2006) or by MCMC sampling (Hoffman et al., 2008). In this paper we address the model-free case, where the MDP dynamics is unknown but the robot can interact with the MDP and collect data. We adopt the probabilistic model of Vlassis and Toussaint (2009) that defines the following joint distribution over random variables R, t, ξ, and $T \in \{0, 1, \ldots, \infty\}$:

$$p(R, t, \xi, T; \theta) = a(T)\ b(t)\ p(\xi|t;\theta)\ p(R|\xi, T), \tag{4}$$

where $p(\xi|t;\theta)$ is defined in (3), and

$$a(t) = (1-\delta)\delta^t, \tag{5}$$

$$b(t) = (1-\gamma/\delta)(\gamma/\delta)^t, \tag{6}$$

are geometric distributions for $t = 0, \ldots, \infty$, with $\gamma < \delta < 1$. The random variable T can be thought of as providing a 'maximal' trajectory length in the sense that, for given T, only trajectories with length shorter than T have nonzero probability when conditioned on the event R. The following theorem shows that the discounted value function $J(\theta)$ of the MDP is proportional to the mixture likelihood $p(R;\theta)$ of the above model:

Theorem 1. *The value function (1) is proportional to the mixture likelihood*

$$J(\theta) \propto \sum_{T=0}^{\infty} a(T) \sum_{t=0}^{\infty} b(t) \int_{\xi \in \mathcal{X}^t} p(\xi|t;\theta)\, p(R|\xi,T), \tag{7}$$

with

$$p(R|\xi,T) = \begin{cases} r_{|\xi|}, & \text{if } |\xi| \leq T \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where $r_{|\xi|}$ is the reward at the last time step of ξ.

Proof. A straightforward adaptation to the continuous setting of the corresponding theorem for discrete state-action spaces (Vlassis and Toussaint, 2009) by replacing the summations over trajectories to integrals. □

Note that the above model is parametrized by a single scalar δ that controls the length of the trajectories. For values of δ close to γ, the value $J(\theta)$ corresponds to the shortest path formulation of an infinite-horizon value function. Contrary to the original probabilistic model of Toussaint and Storkey (2006), the above model allows using all rewards observed in the course of a trajectory during learning (Vlassis and Toussaint, 2009).

3 Likelihood maximization with a Monte Carlo EM algorithm

We want to maximize over θ the function $J(\theta)$ from (7); equivalently we can maximize the log-likelihood function $L(\theta) = \log p(R;\theta)$. We use the EM algorithm for optimization, in which we iteratively maximize an energy function $F(\theta, q)$ that is a lower bound of $L(\theta)$ (Neal and Hinton, 1998). The energy F is a function of the unknown parameters θ and an arbitrary distribution $q \equiv q(\xi, T, t)$ over the 'hidden' variables $\xi \in \mathcal{X}^t$ and $t, T \in \mathbb{N}_0$, defined as follows (the two decompositions are equivalent):

$$F(\theta, q) = L(\theta) - D_{KL}\big[q(\xi,T,t) \;||\; p(\xi,T,t|R;\theta)\big] \tag{9}$$

$$= E_{q(\xi,T,t)}\big[\log p(R,\xi,T,t;\theta)\big] + H(q). \tag{10}$$

In (9) the term D_{KL} is the Kullback-Leibler divergence between $q(\xi, T, t)$ and $p(\xi, T, t|R; \theta)$, the Bayes posterior over t, T and ξ given observed data R and parameters θ. In (10) the first term is the expectation of the joint log-likelihood over the distribution q, and $H(q)$ is the entropy of q (which is independent of θ). We know that the Kullback-Leibler divergence between two distributions is always nonnegative and it becomes zero when the two distributions are equal, hence from (9) we see that F is indeed a lower bound of L, for any choice of q.

An EM algorithm for maximizing F performs coordinate ascent in the space of θ and q: in the E-step we fix θ and maximize F over q; in the M-step we fix q and maximize F over θ. This procedure converges to a local maximum of F (which may also be a local maximum of L). For the E-step, we see from (9) that the optimal distribution q^* that maximizes F is the Bayes posterior computed with parameters θ_{old} found at the last M-step:

$$q^*(\xi, T, t) = p(\xi, T, t|R; \theta_{\text{old}}) \tag{11}$$

$$\propto a(T)\ b(t)\ p(\xi|t; \theta_{\text{old}})\ p(R|\xi, T). \tag{12}$$

In this case holds $F(\theta, q^*) = L(\theta)$ and the two functions touch each other at the value of θ_{old} (and local maxima of F are also local maxima of L). However, other choices of q are also possible as long as we are guaranteed not to decrease F (but in this case local maxima of F may no longer correspond to local maxima of L).

In the M-step we maximize F over θ using the second decomposition (10). When using the optimal q^* from (12), and ignoring terms that do not depend on θ, the energy reads:

$$F(\theta, q^*) = E_{p(\xi,T,t|R;\theta_{\text{old}})}\big[\log p(\xi|t; \theta)\big]. \tag{13}$$

Clearly we cannot evaluate the above expectation analytically when we do not have a model of the MDP, so we have to approximate the expectation by sampling trajectories from the MDP. This is the general idea behind the Monte-Carlo EM algorithm (MCEM) (Wei and Tanner, 1990), that is, replacing a difficult to compute expectation in the E-step by drawing samples from q^*. Here we draw from $q^* \propto a(T)\ b(t)\ p(\xi|t; \theta_{\text{old}})\ p(R|\xi, T)$ by first sampling a maximal length T from $a(T)$, then sampling a T-length trajectory $\xi \sim p(\xi|T; \theta_{\text{old}})$ from the MDP using θ_{old}, and then using all t-length sub-trajectories of ξ, for $t = 0, \ldots, T$, as draws from q^* with importance weights reflecting reward. Assume that m trajectories ξ_i, $i = 1, .., m$, are drawn as described above from $a(T)\ p(\xi|T; \theta_{\text{old}})$; then an estimate of the energy

$F(\theta, q^*)$ in (13) is

$$\hat{F}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|\xi_i| + 1} \sum_{t=0}^{|\xi_i|} w_{it} \ \log p(\xi_i|t;\theta), \tag{14}$$

where $p(\xi_i|t;\theta)$ is the likelihood of the t-length prefix of ξ_i from (3), and w_{it} are importance weights given by

$$w_{it} = b(t) \ r_{it}, \tag{15}$$

where r_{it} is the reward obtained at step t of trajectory ξ_i. (Note that when the model is parametrized by $\delta \approx \gamma$, then $b(t) \approx 1$ and the importance weights are just rewards.) Alternatively one can sample trajectories using a different controller with parameters θ_{other}, in which case the importance weights read:

$$w_{it} = \frac{p(\xi_i;\theta_{\text{old}})}{p(\xi_i;\theta_{\text{other}})} \ b(t) \ r_{it}, \tag{16}$$

and the ratio $p(\xi;\theta_{\text{old}})/p(\xi;\theta_{\text{other}})$ can be directly computed without knowledge of the MDP model (Sutton and Barto, 1998).

When $\log p(\xi|t;\theta)$ is quadratic in the parameters θ (e.g., the case of a linear-Gaussian controller), the maximization of the approximate energy in (14) can be done analytically by solving a weighted least squares problem. We adopt here the same controller parametrization that was used by Kober and Peters (2009) based on a model suggested by Rückstieß et al. (2008), in which the controller is linear-Gaussian with state dependent noise:

$$u_t = (\theta + \varepsilon_t)^\top \phi(x_t), \tag{17}$$

where $\phi : \mathbb{R}^n \mapsto \mathbb{R}^d$ are fixed basis functions and ε_t is white Gaussian noise $\varepsilon_t \sim \mathcal{N}(\varepsilon_t; 0, \sigma^2 I_d)$ that enforces exploration. The variance parameter σ can be either kept fixed or it can be learned in the same framework; here we assume it is fixed without loss of generality. Let ε_{it} be the observed noise term at time step t of a sampled trajectory ξ_i. The contribution of a single ξ_i to the energy $\hat{F}(\theta)$ in (14) is (dropping constant multiplicative and additive

terms):

$$\hat{F}_i(\theta) = \frac{1}{|\xi_i|+1} \sum_{\tau=0}^{|\xi_i|} b(\tau)\ r_{i\tau} \sum_{t=0}^{\tau} \log \pi_\theta(u_{it}|x_{it}) \tag{18}$$

$$= \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} \Big[\sum_{\tau=t}^{|\xi_i|} b(\tau)\ r_{i\tau} \Big] \log \pi_\theta(u_{it}|x_{it}) \tag{19}$$

$$= \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} Q_{it} \log \pi_\theta(u_{it}|x_{it}), \tag{20}$$

where we defined

$$Q_{it} = \sum_{\tau=t}^{|\xi_i|} b(\tau)\ r_{i\tau}. \tag{21}$$

Differentiating (20) over θ and neglecting terms independent of θ gives (in the one-dimensional case $d = 1$)

$$\theta^* = \theta_{\text{old}} + \frac{\sum_{i=1}^{m} (1+|\xi_i|)^{-1} \sum_{t=0}^{|\xi_i|} Q_{it} \varepsilon_{it}}{\sum_{i=1}^{m} (1+|\xi_i|)^{-1} \sum_{t=0}^{|\xi_i|} Q_{it}}, \tag{22}$$

and more generally

$$\theta^* = \theta_{\text{old}} + \Big\{ \sum_{i=1}^{m} (1+|\xi_i|)^{-1} \sum_{t=0}^{|\xi_i|} Q_{it} W_{it} \Big\}^{-1} \times \Big\{ \sum_{i=1}^{m} (1+|\xi_i|)^{-1} \sum_{t=0}^{|\xi_i|} Q_{it} W_{it} \varepsilon_{it} \Big\}, \tag{23}$$

where $W_{it} = \phi(x_{it})\phi(x_{it})^\top / (\phi(x_{it})^\top \phi(x_{it}))$.

4 The episodic case: The PoWER algorithm

Here we show that the PoWER algorithm of Kober and Peters (2009) corresponds to a special case of our model. PoWER applies on episodic tasks where each sampled trajectory has fixed length H, and the goal is to maximize the non-discounted value function[1]

$$J(\theta) = E\Big[\sum_{t=0}^{H} r_t; \theta \Big]. \tag{24}$$

[1]The discounted finite-horizon case was also addressed recently (Peters and Kober, 2009).

It is straightforward to see that our model (4) can still be used in this case, by assuming a degenerate distribution $a(T)$ where $a(T) = 1$ for $T = H$ and 0 otherwise, and a uniform distribution $b(t) = \frac{1}{1+H}$, for $T = 0, 1, \ldots, H$. The update equations in the M-step of EM are then identical to (22) and (23), with the simplification that each trajectory has identical length H and the quantities $b(t)$ are constant. One then gets precisely the update equations of PoWER, with $Q_{it} = \sum_{\tau=t}^{H} r_{i\tau}$.

Comparing the two algorithms, MCEM and PoWER, we note two main differences. The first is that MCEM has a built-in capacity to handle discounted infinite horizon problems by explicitly incorporating the discount factor γ in its update steps (via the terms $b(t)$). This may not necessarily imply, however, that PoWER cannot handle infinite horizon tasks (or the related stochastic shortest path problems): in several problems, fixing H to some constant value allows PoWER to learn a controller with high discounted value. We have observed this in practice in all problems that we tried PoWER: there was always a good ('optimal') value of H for which PoWER would improve the value function, even in infinite horizon problems. However this value of H was hard to locate: different problems required different values of H, and even the same problem required different values of H depending on the initialization of the parameters θ. On the contrary, MCEM does not require setting a fixed value for H, as the length of each sampled trajectory is a random draw from the geometric distribution $a(T) \propto \delta^T$. Effectively MCEM 'randomizes' over H in each step, and provides an insightful way to generalize PoWER to the discounted setting.

The second difference between MCEM and PoWER is that in the update equations of MCEM the rewards are discounted by the quantities $b(t)$, which are decaying in t. This can have a positive effect on the convergence of the MCEM algorithm when the MDP dynamics is noisy: the algorithm weighs early rewards more heavily than rewards arriving later in a trajectory, as the latter could have been contaminated by noise in the trajectory dynamics. We note however that in recent work, Peters and Kober (2009) have extended PoWER to incorporate discounting (based on γ) in the update equations, similar to MCEM.

5 Experiments

In this section we summarize results from some preliminary experiments on two simulated problems and a real robot balancing problem. In the simulated problems we only compare MCEM to the PoWER algorithm, as the

main focus of this work was to extend the PoWER algorithm to the discounted infinite horizon setting, so it is interesting to see experimentally the differences between these two algorithms. (The theoretical and experimental comparison of MCEM to other methods in the literature, both model-free and model-based, is a matter of future work.) In the real robot problem we only tried MCEM, using importance sampling in order to reduce sample complexity as we explain below.

The first synthetic problem is a two-dimensional MDP with nonlinear dynamics, similar to the one used by Toussaint and Storkey (2006), for which the value function $J(\theta)$ as a function of $\theta \in \mathbb{R}^2$ exhibits a single maximum (we verified this by estimating $J(\theta)$ over a grid of values for θ). The state space is $x = [x_1, x_2]$, where x_1 is robot position and x_2 is robot velocity, the control action u produces acceleration, and the MDP dynamics is

$$x_2(t+1) = x_2(t) + \frac{1}{1+\exp(-u(t))} - 0.5 + \kappa, \tag{25}$$

$$x_1(t+1) = x_1(t) - 0.1x_2(t+1) + \kappa, \tag{26}$$

where κ is zero-mean Gaussian noise with $\sigma_\kappa = 0.02$. The robot starts from position $x_1 = 1$ (plus some Gaussian noise with standard deviation 0.001) and velocity $x_2 = 0$, and must reach the state $[0, 0]$. The discount factor is $\gamma = 0.95$. The reward is modeled as a rare event, $r(t) = 1$ if $||x|| \leq 0.1$ and 0 otherwise. The control policy is stochastic and time-invariant, given by $u_t = (\theta + \varepsilon_t)^\top x_t$, with Gaussian exploration noise $\varepsilon_t \sim \mathcal{N}(\varepsilon_t; 0, \sigma_\varepsilon^2 I_2)$ with fixed $\sigma_\varepsilon = 0.5$, where the choice of σ_ε was guided by the dimensions of the parameter landscape (the value function is nontrivial approximately for $\theta_1 \in [0, 60]$ and $\theta_2 \in [-8, 0]$).

We applied MCEM and PoWER starting from random values of θ within the nontrivial region of the parameter landscape, and report average results. In each EM iteration we sampled 50 trajectories. In MCEM we used $\delta = 0.99$, and in PoWER we used horizon length $H = 80$. The value of H in PoWER was chosen in such a way that the algorithm would reach the (rare) reward as fast as possible for any initial value of θ. (Note that this is a near-optimal way to choose H; in a real-world problem there is no obvious way to choose a single good value of H). In each EM iteration we computed the discounted cumulative reward of each controller by sampling 500 trajectories through the MDP. In Fig. 1 we see the results, where the x-axis shows the iterations of EM, and the y-axis shows discounted cumulative reward for each value of θ. This experiment highlights the advantage of MCEM over PoWER in the way the two algorithms handle the horizon depth: by randomizing

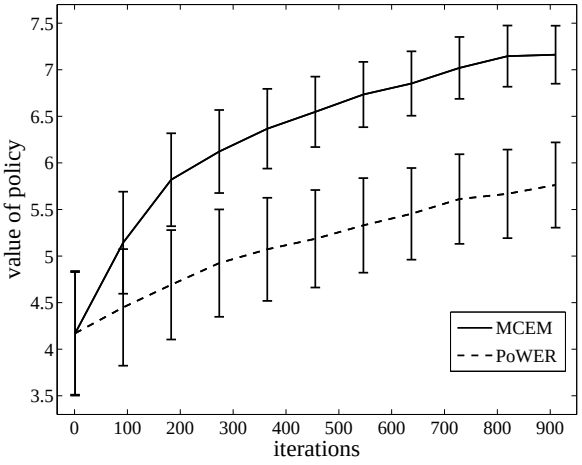

Figure 1: MCEM vs PoWER on a synthetic two-dimensional MDP with nonlinear dynamics. The x-axis indicates the EM iterations, and the y-axis shows the estimated discounted value of each controller. For MCEM we used $\delta = 0.99$, and for PoWER $H = 80$.

over the length of each sampled trajectory, MCEM is able to locate the rare event more often than PoWER on the average, both at the initial stages of the algorithm where θ is far from the optimal value and hence large trajectories are needed to collect reward, as well as at later stages of the algorithm (near the optimal θ) when only a few steps are needed to collect reward.

The second simulated experiment is a helicopter hovering problem involving high-dimensional, noisy, nonlinear dynamics. For this experiment we used a helicopter simulator provided by Pieter Abbeel, Adam Coates and Andrew Ng at the Stanford University, and which has been used at the 2009 Reinforcement Learning Competition as a competition domain.[2] The state space here is 9-dimensional, involving the helicopter's position (x, y, z), orientation (roll ϕ, pitch θ, yaw ω) and velocity $(\dot{x}, \dot{y}, \dot{z})$. The action space is 4-dimensional consisting of:

- u_1: The longitudinal (front-back)cyclic pitch control.
- u_2: The latitudinal (left-right) cyclic pitch control.
- u_3: The main rotor collective pitch control.

[2]See www.rl-competition.org

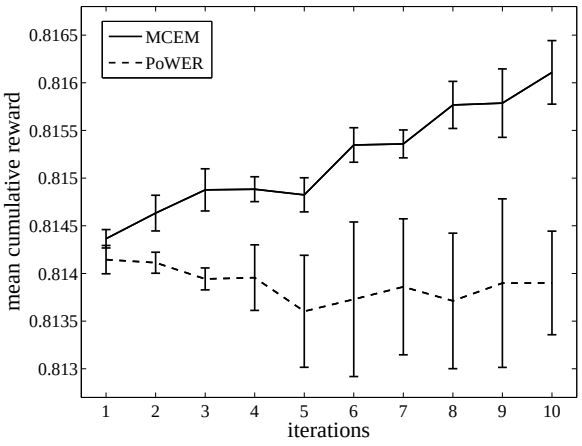

Figure 2: MCEM vs PoWER in the simulated helicopter hovering problem. The x-axis indicates the EM iterations, and the y-axis shows average cumulative reward.

- u_4: The tail rotor collective pitch control.

The task here is to hover the helicopter based on feedback we are given in each time step, that comes in the form of a 9-dimensional error vector that includes the distance of each of the state parameters from a target point in the parameter space. Based on this error vector we defined the following controller:

$$u_1 = -w_1 x_{error} - w_2 \dot{x}_{error} - w_3 \theta_{error} + w_4, \tag{27}$$

$$u_2 = -w_5 y_{error} - w_6 \dot{y}_{error} - w_7 \phi_{error} + w_8, \tag{28}$$

$$u_3 = w_9 z_{error} - w_{10} \dot{z}_{error} + w_{11}, \tag{29}$$

$$u_4 = -w_{12} \omega_{error}, \tag{30}$$

parametrized by the 9-dimensional vector w. (The parameters w_4, w_8 and w_{11} are constant terms used for the helicopter trimming.)

We tested PoWER and MCEM algorithms in this benchmark. For PoWER we chose $H = 50$ (after tuning) and for MCEM we chose $\delta = 0.99$ (and assumed $\gamma = 0.95$). Each trajectory had maximal allowed length 6000 time steps, which translates into 10 minutes of flying time in the real helicopter. In each EM iteration we used batches of 100 trajectories, and set the exploration noise equal in both algorithms. In each test run we initialized the parameters by manual tuning based on software provided by the

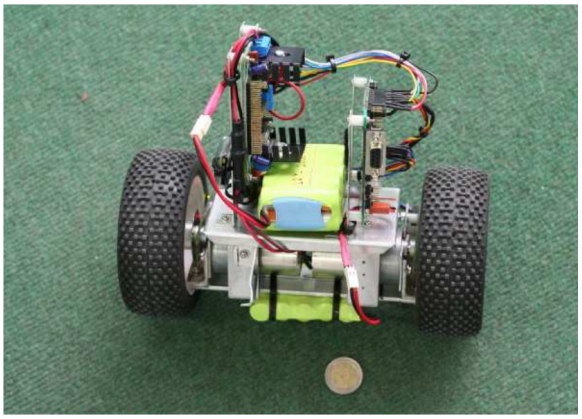

Figure 3: Our self-balancing robot Robba.

RL Competition. The evaluation was performed according to the Reinforcement Learning Competition rules, being the average of the rewards collected during the learning phase. The reward in each time step was calculated in both algorithms as follows:

$$r(t) = \exp(-x_{error}^2) + \exp(-\dot{x}_{error}^2) + \exp(-\theta_{error}^2) + \exp(-y_{error}^2) + \exp(-\dot{y}_{error}^2) + \exp(-\phi_{error}^2) + \exp(-z_{error}^2) + \exp(-\dot{z}_{error}^2) + \exp(-\omega_{error}^2). \quad (31)$$

If the helicopter's state exceeds some error bounds, the helicopter crashes. In our implementations neither of the algorithms ever crashed the helicopter. In Fig. 2 we see the results of learning for MCEM and PoWER, from which similar conclusions as in the toy 2d problem can be drawn, namely that the MCEM algorithm allows more flexibility by working with a large enough δ, obviating the need to define a precise horizon depth H as in PoWER.

The third experiment involves a real balancing robot. We have built our own wheeled balancing robot, called Robba, which is shown in Fig. 3. Robba uses differential drive and it was explicitly designed to become a compact, low cost self-balancing robot. The vehicle comprises an aluminum frame that supports the following components:

- Two 12 Vdc, 152 RpM spur geared motors
- An ooPIC microcontroller with the R carrier board

- One dual PWM motor driver
- Two 64 pulses per revolution odometers, one for each wheel
- A dual axis accelerometer (ADXL203) and a single axis gyro (ADXRS300), both from Analog Devices
- Two energy sources, a 12V 2700mAh rechargeable battery for the motors and a 6V 2700mAh rechargeable battery for the electronics

One of the main considerations for the vehicle design was the achievement of compact dimensions and robust construction, able to confront the strains during the learning period. Robba uses two 12cm diameter wheels, it is 12cm long, 24cm wide with a height of 21cm and weighs 2Kg.

In our learning experiment we start the robot from zero vertical angle plus some noise, and we instantly give large equal torque to both motors resulting in loss of balance for the robot. Our goal is to have the robot recover its initial vertical position as fast as possible and stay in balance. The state space here is four-dimensional, involving vertical angle x_1, angular velocity x_2, linear velocity x_3, and position on the horizontal axis x_4. The torque control is given by $u = (\theta + \varepsilon)^T x$, where $x = [x_1, x_2, x_3, x_4]^\top$ and $\theta \in \mathbb{R}^4$, and ε_t is exploration noise. In each step we penalize any angle different to 0 degrees and any linear speed of the robot using rewards

$$r(t) = \frac{1}{2}\exp(-x_1^2(t)) + \frac{1}{2}\exp(-x_3^2(t)). \tag{32}$$

In order to have a good estimate for the angle of the robot, we fused the measurements of the gyro and the accelerometer using a simple Kalman filter. In each run we initialized the parameters by manual tuning and in accordance to analytical model-based solutions of similar systems (Kim et al., 2005). In each episode we sampled batches of 20 trajectories. In this experiment we only used the MCEM algorithm with $\gamma = 0.99$ and $\delta = 0.992$, and we used importance sampling of trajectories as in (16). After the learning phase we evaluated each deterministic policy by trying it five times on the real robot with trajectory length 200 time steps. The learning curve of MCEM is shown in Fig. 4. After learning the robot was able to recover fast from the initial disturbance (as well as from subsequent similar disturbances that we manually created) and stabilize its position at zero angle. Note that the use of importance sampling resulted in significant speedups, as only five iterations of EM were sufficient to achieve good balancing behavior.

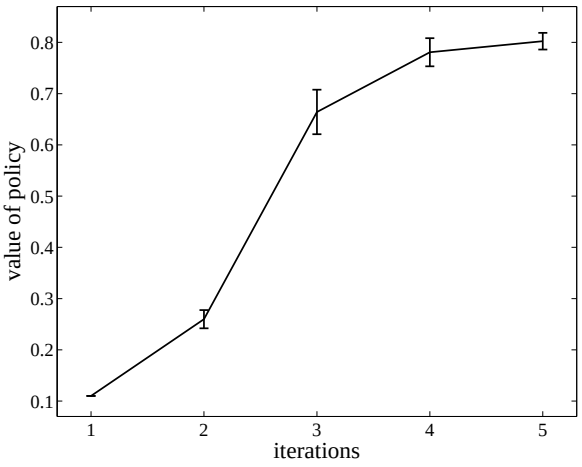

Figure 4: Learning curve of Robba with MCEM and importance sampling. The x-axis indicates the EM iterations, and the y-axis shows estimated discounted value.

6 Conclusions

We proposed a Monte Carlo EM algorithm (MCEM) for learning model-free robot control that is based on the probabilistic model of Vlassis and Toussaint (2009) for model-free Reinforcement Learning. The MCEM algorithm searches in the space of controller parameters using information obtained from randomly generated robot trajectories, and it can be viewed as the generalization of the PoWER algorithm of Kober and Peters (2008) to the discounted infinite-horizon case. An interesting aspect of our framework is that it allows treating the infinite-horizon case as a 'randomized' version of the episodic case, whereby the length of each trajectory is a random sample from a geometric distribution in an appropriately constructed probabilistic model. We have provided some preliminary experiments comparing MCEM to PoWER in two simulated and one real-world robotic problems, highlighting the potential of MCEM to handle infinite horizon problems. Ongoing work aims at studying the relationships of MCEM to state-of-the-art policy gradient algorithms for robotics (Peters and Schaal, 2008a,b).

Acknowledgements

We give warm thanks to Jan Peters for various helpful discussions.

References

Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In *Proc. Neural Information Processing Systems*.

Bertsekas, D. P., and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. *Proc. 4th Workshop on Uncertainty in Artificial Intelligence* (pp. 55–63). Minneapolis, Minnesota, USA.

Dayan, P. and Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1–38.

Hoffman, M., Doucet, A., de Freitas, N., and Jasra, A. (2008). Bayesian policy learning with trans-dimensional MCMC. In *Proc. Advances in Neural Information Processing Systems 20*.

Kim, Y. and Kim, S. H. and Kwak, Y. K. (2005). Dynamic Analysis of a Nonholonomic Two-Wheeled Inverted Pendulum Robot. *Journal of Intelligent and Robotic Systems*, 44(1):25–46.

Kober, J. and Peters, J. (2009). Policy search for motor primitives in robotics. In *Proc. Advances in Neural Information Processing Systems 21*.

Martinez-Cantin, R. and de Freitas, N. and Castellanos, J. A. and Doucet, A. (2009). A Bayesian Exploration-Exploitation Approach for Optimal Online Sensing and Planning with a Visually Guided Mobile Robot. *Autonomous Robots*, 27(1). This issue, part B.

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in graphical models*, pages 355–368. Kluwer Academic Publishers.

Ng, A. Y. and Jordan, M. I. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Proc. Uncertainty in Artificial Intelligence*.

Peters, J. and Kober, J. (2009). Using Reward-Weighted Imitation for Robot Reinforcement Learning. In *Proc. 2009 IEEE Int. Symp. on Approximate Dynamic Programming and Reinforcement Learning*.

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

Peters, J., and Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682-697.

Riedmiller, M. and Gabel, T. and Hafner, R. and Lange, S. (2009). Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73. This issue, part A.

Rückstieß, T. and Felder, M. and Schmidhuber, J. (2008). State-Dependent Exploration for Policy Gradient Methods. In *Proc. European Conf. on Machine Learning*.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tedrake, R., Zhang, T. W., and Seung, H. S. (2005). Learning to walk in 20 minutes. In *Proc. 14th Yale Workshop on Adaptive and Learning Systems*.

Toussaint, M. and Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proc. Int. Conf. on Machine Learning*.

Vlassis, N. and Toussaint, M. (2009). Model-free Reinforcement Learning as Mixture Learning. In *Proc. Int. Conf. on Machine Learning*. Montreal, Canada.

Wei, G. and Tanner, M. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithm. *J. Amer. Statist. Assoc.*, 85:699–704.