Model-Free Reinforcement Learning as Mixture Learning

Nikos Vlassis VLASSIS@DPEM.TUC.GR
Technical University of Crete, Dept. of Production Engineering and Management, 73100 Chania, Greece

Marc Toussaint MTOUSSAI@CS.TU-BERLIN.DE
TU Berlin, Franklinstr 28/29 FR6-9, 10587 Berlin, Germany

Abstract

We cast model-free reinforcement learning as the problem of maximizing the likelihood of a probabilistic mixture model via sampling, addressing both the infinite and finite horizon cases. We describe a Stochastic Approximation EM algorithm for likelihood maximization that, in the tabular case, is equivalent to a non-bootstrapping optimistic policy iteration algorithm like Sarsa(1) that can be applied both in MDPs and POMDPs. On the theoretical side, by relating the proposed stochastic EM algorithm to the family of optimistic policy iteration algorithms, we provide new tools that permit the design and analysis of algorithms in that family. On the practical side, preliminary experiments on a POMDP problem demonstrated encouraging results.

1. Introduction

Reinforcement Learning (RL) is the problem of learning to control a stochastic dynamical system (or an agent interacting with its environment) by simulation and trial-and-error (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). RL methods hold great promise in learning controllers for systems with unknown dynamics, and they have recently demonstrated impressive results (Abbeel et al., 2007; Kober & Peters, 2009).

In this paper we describe a reduction of model-free RL to a problem of likelihood maximization in a mixture model. Our approach is based on the work of Toussaint and Storkey (2006) who showed that the value function of a known MDP is proportional to the likelihood function of an appropriately constructed mixture model. This likelihood can then be maximized by probabilistic inference techniques like EM or Markov Chain Monte Carlo (Hoffman et al., 2008). We show in this paper that when the model of the MDP is unavailable we can use a stochastic EM algorithm to optimize the policy based on trajectory samples only. In particular, we use the Stochastic Approximation EM (SAEM, Delyon et al., 1999) where the E-step is performed by sampling from the MDP with importance weights that reflect rewards. However, a direct translation of the model of Toussaint and Storkey (2006) to model-free RL leads to an inefficient sampling algorithm that uses only the reward at the last time step of a sampled trajectory. We propose an alternative formulation that ensures that all rewards observed along a trajectory are taken into account in the mixture learning.

In the tabular case (discrete states and actions), the proposed SAEM algorithm turns out to be identical to a non-bootstrapping optimistic policy iteration algorithm similar to Sarsa(1). Optimistic policy iteration (OPI) is an instance of approximate policy iteration where the policy is updated before it is completely evaluated (Bertsekas & Tsitsiklis, 1996; Tsitsiklis, 2002). In OPI, one or more complete or partial trajectories are generated according to the current policy, the observed rewards are then used to update that policy, and so forth. Of particular interest are non-bootstrapping OPI methods that are based on Monte-Carlo partial policy evaluation, because these methods can be directly applied to partially observable domains (POMDPs) as well.

The convergence of OPI is not always guaranteed, even for tabular MDPs, and it depends on various factors including the choice of trajectories for simulation, the amount of bootstrapping used, the frequency of policy updates, and the particular form of the policy update operator (Tsitsiklis, 2002). Several counter-examples

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

have been given in the literature that demonstrate possible non-convergent behavior of OPI (Bertsekas & Tsitsiklis, 1996; Gordon, 1996). For application of OPI methods in POMDPs the theoretical results are scarce. If the policy update operator is a sufficiently smooth function of the action values, then we know that value and policy fixed points do exist (Perkins & Pendrith, 2002), and convergence to a fixed point is guaranteed if the policy is fully evaluated before it is updated (Perkins & Precup, 2003). A similar result has been established by Melo et al. (2008). However, the general convergence of non-boostrapping OPI algorithms is still open.

By relating the proposed SAEM algorithm to the family of OPI algorithms like Sarsa, we provide new tools that permit the design and analysis of OPI algorithms. We report some preliminary experiments in which SAEM demonstrates encouraging results on a standard POMDP problem.

2. Value function as mixture likelihood

Consider an infinite-horizon MDP on the random variables of state x_t and action u_t as defined by the start distribution $p(x_0)$, stationary transition probability $p(x_{t+1} \,|\, u_t, x_t)$, and a deterministic reward signal $r_t \equiv r(u_t, x_t)$. Given a stochastic policy $\pi(u_t|x_t)$ we define the value $V(\pi)$ as the expected discounted future reward:

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

where $\gamma \in (0,1)$ is a known discount factor. Our task is to find a stationary policy π that maximizes $V(\pi)$.

Toussaint and Storkey (2006) showed that the value function can be expressed as a likelihood in an infinite-component mixture model. To express this formally we use the notation $\xi = (x_0, u_0, .., x_T, u_T)$ for a state-action trajectory of length $|\xi| = T$. Let $\mathcal{X}^T = \{\xi : |\xi| = T\}$ be the space of length-T trajectories and $\mathcal{X}^* = \bigcup_{T=0}^{\infty} \mathcal{X}^T$ the space of arbitrary length trajectories. For a given MDP and policy π we define the joint probability distribution over $\xi \in \mathcal{X}^*$ and T as,

$$p(\xi, T; \pi) = \pi(u_T|x_T) \Big[\prod_{t=0}^{T-1} p(x_{t+1}|u_t, x_t)\ \pi(u_t|x_t)\Big] \cdot p(x_0)\ \delta_{|\xi|T}\ p(T)\ . \tag{2}$$

Here, $\delta_{|\xi|T}$ is one for $|\xi| = T$ and zero otherwise, and $p(T)$ is a prior over the trajectory length. From this joint probability distribution we can define a conditional distribution over $\xi \in \mathcal{X}^*$

$$p(\xi|T; \pi) = \frac{p(\xi, T; \pi)}{p(T)}, \tag{3}$$

and a marginal distribution over $\xi \in \mathcal{X}^*$

$$p(\xi; \pi) = \sum_{T=0}^{\infty} p(\xi|T; \pi) p(T). \tag{4}$$

Note that both are properly normalized over the whole space $\xi \in \mathcal{X}^*$. When rewards are scaled to $[0, 1]$ we can introduce an auxiliary binary random variable R with

$$p(R{=}1 \,|\, u_T, x_T) = r(u_T, x_T), \tag{5}$$

an idea suggested earlier in other contexts (Cooper, 1988; Dayan & Hinton, 1997; Peters & Schaal, 2008). If we further define

$$p(R{=}1|\xi) = p(R{=}1|u_{|\xi|}, x_{|\xi|}) \tag{6}$$

we get the following joint distribution over R and $\xi \in \mathcal{X}^*$:

$$p(R, \xi; \pi) = p(R|\xi)\ p(\xi; \pi). \tag{7}$$

If we assume a geometric prior $p(t) = (1-\gamma)\gamma^t$ then we can rewrite the value function (1) as

$$\begin{aligned} V(\pi) &= \sum_{T=0}^{\infty} \gamma^T E\big[r_T; \pi\big] \\ &= \sum_{T=0}^{\infty} \gamma^T \sum_{u_T, x_T} p(u_T, x_T; \pi)\ r(u_T, x_T) \\ &= \frac{1}{1-\gamma} \sum_{T=0}^{\infty} p(T) \sum_{\xi \in \mathcal{X}^T} p(\xi|T; \pi)\ p(R{=}1|\xi) \\ &= \frac{1}{1-\gamma} \sum_{\xi \in \mathcal{X}^*} p(\xi; \pi)\ p(R{=}1|\xi). \end{aligned} \tag{8}$$

That is, we have related the value function $V(\pi)$ to the mixture likelihood $p(R{=}1; \pi)$ in the joint distribution $p(R, \xi; \pi)$, where the variable-length trajectory $\xi \in \mathcal{X}^*$ is the latent variable. The model (8) can also be viewed as a generative process for the auxiliary random variable R: first sample a time step T from $p(T)$, then sample a length-T trajectory $\xi \sim p(\xi|T; \pi)$ from the MDP following policy π, and finally sample the binary random variable R from $p(R|u_T, x_T)$. The probability that $R = 1$ is then proportional to the value $V(\pi)$.

Modeling the value function as a mixture likelihood offers the possibility to use probabilistic inference techniques for optimization, like the EM algorithm (Toussaint & Storkey, 2006) or Markov Chain Monte

Carlo (Hoffman et al., 2008). In particular, in the EM algorithm we compute a parameter π that locally maximizes the likelihood $p(R;\pi)$ by iteratively maximizing a lower bound of $\log p(R;\pi)$ (Neal & Hinton, 1998). Let $q(\xi)$ be a distribution over the latent variable $\xi \in \mathcal{X}^*$. Consider the function

$$F(q,\pi) = \log p(R;\pi) - D\big(q(\xi) \,\|\, p(\xi|R;\pi)\big) \quad (9)$$

$$= \sum_{\xi\in\mathcal{X}^*} q(\xi)\Big[\log p(R|\xi) + \log p(\xi;\pi)\Big] + H(q), \quad (10)$$

where $D(\cdot)$ is the Kullback-Leibler divergence between two distributions, and $H(\cdot)$ is the entropy of a distribution. We start with an initial guess of the parameter π, and then we alternate between an E-step and an M-step. In the E-step we find a distribution q that maximizes $F(q,\pi)$ for fixed π, using (9). The optimal q that minimizes the Kullback-Leibler divergence in (9) is the posterior over the latent variables given the observed data (in our case the hypothetical $R = 1$) and for given parameters π_{old}:

$$q^*(\xi) = p(\xi \,|\, R;\pi_{\text{old}}). \quad (11)$$

In the M-step we find a new parameter π that maximizes $F(q^*,\pi)$ for the optimal q^*, using (10). Often this maximization can be performed analytically.

If the model of the MDP is available then the EM algorithm is similar to ordinary policy iteration: the E-step involves a full sweep of the state-action space in order to compute q^*, which is analogous to policy evaluation. The M-step maximizes $F(q^*,\pi)$ over the policy parameters π, which is analogous to policy improvement. In this paper we are interested in the model-free case, in which we do not have access to the transition model of the MDP. In this case we cannot perform exact inference in the E-step, and hence we have to resort to sampling, giving rise to a stochastic EM algorithm.

3. Stochastic Approximation EM

In the model-free case we propose to use a stochastic EM algorithm in which the E-step posterior distribution is estimated by Monte Carlo sampling from the MDP, and hence no explicit knowledge of the MDP model is needed. There are several variants of stochastic EM in the literature that differ in the details of the Monte Carlo approximation, particularly in the number of Monte Carlo draws per iteration, and in their conditions for convergence (Celeux & Diebolt, 1985; Wei & Tanner, 1990; Delyon et al., 1999). In our work we use the Stochastic Approximation EM algorithm (SAEM, Delyon et al., 1999) that is easy to implement and guarantees convergence to a local maximum of the likelihood function under reasonable conditions. An attractive feature of SAEM is that convergence is guaranteed even when the sample size is kept constant over successive iterations of the algorithm, as opposed to other stochastic EM versions that may need to tune the sample size in order to ensure convergence. Additionally, in the context of our RL application the SAEM algorithm can be given a familiar interpretation in terms of state-action values, as we will see in later sections.

The SAEM algorithm iterates between three steps (Delyon et al., 1999): In iteration k we

1. generate m samples ξ_i, $i = 1,\dots,m$, from $p(\xi \,|\, R;\pi_{k-1})$, where π_{k-1} is the parameter from the previous iteration,
2. update a function $\hat{F}_k(\pi)$ according to

$$\hat{F}_k(\pi) = (1-\alpha_k)\hat{F}_{k\text{-}1}(\pi) + \frac{\alpha_k}{m}\sum_{i=1}^{m}\log p(\xi_i;\pi), \quad (12)$$

3. and assign new parameters

$$\pi_k = \operatorname*{argmax}_{\pi} \hat{F}_k(\pi). \quad (13)$$

To understand this scheme, first assume that $\alpha_k = 1$ in each iteration. Then $\hat{F}_k = \frac{1}{m}\sum_{i=1}^{m}\log p(\xi_i;\pi)$ is simply the sample based approximation of the function $F(q,\pi)$ in (10) (neglecting terms independent of π) and the 3rd step is the standard M-step based on this approximation. Special about the SAEM algorithm is that it uses a learning rate α_k, with $0 < \alpha_k < 1$, and thereby *smoothes* the approximation of $F(q,\pi)$ over several iterations, which is a key for its convergence.

In the case of our model $p(R,\xi;\pi)$ (equation (7)) we can use importance weights for the sampling in step 1: we can generate samples from

$$p(\xi \,|\, R{=}1;\pi_{k-1}) \propto p(\xi;\pi_{k-1})\; p(R{=}1|\xi) \quad (14)$$

by first sampling m variable-length trajectories $\xi_i \sim p(\xi;\pi_{k-1})$ from the MDP using the previous policy π_{k-1}, and then assigning them importance weights

$$w_{\xi_i} = p(R{=}1|\xi_i). \quad (15)$$

Note that in this case the importance weights correspond to the terminal rewards $r(u_{|\xi_i|}, x_{|\xi_i|})$ for each trajectory ξ_i. Accordingly, the function $\hat{F}_k$ in step 2 reads

$$\hat{F}_k(\pi) = (1-\alpha_k)\hat{F}_{k\text{-}1}(\pi) + \frac{\alpha_k}{m}\sum_{i=1}^{m} w_{\xi_i}\log p(\xi_i;\pi). \quad (16)$$

For particular choice of policy parameters π, the function $\hat{F}_k$ can be maximized analytically as we will see later on.

An advantage of SAEM over other stochastic EM algorithms is that step 2 is effectively making use of *all* simulated data when optimizing for the current π_k. Translated in our RL problem, this means that the optimal policy π_k at step k depends on trajectories and rewards observed when following policies other than π_k at earlier steps. This feature makes SAEM similar to on-policy control algorithms like Sarsa (Sutton & Barto, 1998) that iteratively estimate Q-functions using rewards obtained by following previous policies. The SAEM algorithm is guaranteed to converge to a local maximum of the likelihood function for models in the exponential family, under general conditions that include the standard conditions of stochastic approximation for α_k (Delyon et al., 1999).

However, a drawback of using the model $p(R, \xi; \pi)$ in (7) is that the function $\hat{F}_k(\pi)$ in (16) is making use only of a single reward per trajectory (the terminal one), and therefore a lot of observed rewards are wasted. The result is that a huge set of trajectories would be needed in order for $\hat{F}_k(\pi)$ to contain useful information for computing the new π_k. Although the SAEM framework described above justifies the use of a single reward per trajectory for deriving a convergent on-policy control algorithm, such an algorithm would be of little practical significance as it would require an unrealistically large data sample for producing meaningful results. (We have empirically verified this in practice.) In the next section we show that we can express the value function using a different mixture model, in which the M-step of the corresponding SAEM algorithm uses all intermediate rewards in a trajectory.

4. An alternative mixture model for V

We show here that the value function $V(\pi)$ can be expressed in terms of an alternative mixture model:

Theorem 1. *There exists a parametrized family of geometric distributions $a(t)$ and $b(t)$ for which the value function is proportional to*

$$V(\pi) \propto \sum_{T=0}^{\infty} a(T) \sum_{t=0}^{\infty} b(t) \sum_{\xi \in \mathcal{X}^t} p(\xi|t;\pi)\ p(R{=}1|\xi, T), \tag{17}$$

with

$$p(R{=}1|\xi, T) = \begin{cases} p(R{=}1|u_{|\xi|}, x_{|\xi|}) & \text{if } |\xi| \leq T \\ 0 & \text{otherwise} \end{cases}. \tag{18}$$

Proof. We prove the theorem by explicitly constructing the parametrized family of distributions $a(\cdot)$ and $b(\cdot)$. The family will be parametrized by a scalar δ with $\gamma < \delta < 1$. We define

$$a(t) = (1-\delta)\delta^t, \tag{19}$$

$$b(t) = (1-\gamma/\delta)(\gamma/\delta)^t, \tag{20}$$

for $t = 0, \ldots, \infty$. Since $p(R{=}1|\xi, T) = 0$ for $|\xi| > T$, the right hand side of (17) reads:

$$\sum_{T=0}^{\infty} a(T) \sum_{t=0}^{T} b(t) \sum_{\xi \in \mathcal{X}^t} p(\xi|t;\pi)\ p(R{=}1|u_t, x_t), \tag{21}$$

where we have truncated the t-summation at time T. The term $\sum_{\xi \in \mathcal{X}^t} p(\xi|t;\pi)\ p(R{=}1|u_t, x_t)$ is by definition the expectation $E[r_t; \pi]$ (as in (8)), and the right hand side of (17) reads

$$\sum_{T=0}^{\infty} a(T) \sum_{t=0}^{T} b(t)\ E[r_t;\pi] \tag{22}$$

$$= (1-\delta)(1-\gamma/\delta) \sum_{T=0}^{\infty} \delta^T \sum_{t=0}^{T} (\gamma/\delta)^t E[r_t;\pi] \tag{23}$$

$$= (1-\delta)(1-\gamma/\delta) \sum_{t=0}^{\infty} (\gamma/\delta)^t E[r_t;\pi] \sum_{T=t}^{\infty} \delta^T \tag{24}$$

$$= (1-\delta)(1-\gamma/\delta) \sum_{t=0}^{\infty} (\gamma/\delta)^t E[r_t;\pi]\ \delta^t \sum_{T=0}^{\infty} \delta^T \tag{25}$$

$$= (1-\gamma/\delta) \sum_{t=0}^{\infty} \gamma^t E[r_t;\pi] = (1-\gamma/\delta) V(\pi), \tag{26}$$

and hence (17) holds with proportionality constant $1/(1-\gamma/\delta)$. □

Let us contrast this to the previous mixture model (equations (7) and (8)). In that model we had the joint $p(R, \xi; \pi)$ over R and $\xi \in \mathcal{X}^*$, where the length of ξ was distributed according to $p(T) \propto \gamma^T$ (equation (4)). What we have derived in the above theorem corresponds to a joint distribution over R, t, $\xi \in \mathcal{X}^t$, and a new random variable T:

$$p(R, t, \xi, T; \pi) = a(T)\ b(t)\ p(\xi|t;\pi)\ p(R|\xi, T), \tag{27}$$

with $p(\xi|t;\pi)$ defined in accordance to (3). The new 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 $R{=}1$. Effectively, the T variable implies an additional discounting so that the geometric priors $a(T)$ and $b(t)$ together induce the

appropriate discounting. As the theorem shows, the mixture likelihood $p(R=1)$ is also in the new model proportional to the value function $V(\pi)$.

When $\delta \to \gamma$ (but strictly $\delta > \gamma$) the distribution $b(t)$ is nearly constant (decays very slowly) and the distribution $a(T)$ approximates $p(T)$. In this case, using expression (21), we have

$$\begin{aligned} V(\pi) &\propto \sum_{T=0}^{\infty} a(T) \sum_{t=0}^{T} b(t) \sum_{\xi \in \mathcal{X}^t} p(\xi|t;\pi)\ p(R{=}1|u_t, x_t) \\ &= \sum_{T=0}^{\infty} a(T) \sum_{t=0}^{T} b(t) \sum_{\xi \in \mathcal{X}^T} p(\xi|T;\pi)\ p(R{=}1|u_t, x_t) \\ &= \sum_{T=0}^{\infty} a(T) \sum_{\xi \in \mathcal{X}^T} p(\xi|T;\pi) \sum_{t=0}^{T} b(t)\ p(R{=}1|u_t, x_t) \\ &\approx \sum_{T=0}^{\infty} p(T) \sum_{\xi \in \mathcal{X}^T} p(\xi|T;\pi) \sum_{t=0}^{T} p(R{=}1|u_t, x_t). \quad (28) \end{aligned}$$

In the second line we could extend the summation over $\xi \in \mathcal{X}^t$ to $\xi \in \mathcal{X}^T$ since the additional summation over $(x_{t+1:T}, u_{t+1:T})$ sums to 1. Equation (28) corresponds to the stochastic shortest path formulation of an infinite-horizon value function (Bertsekas & Tsitsiklis, 1996, p. 39), and our theorem can be regarded as a generalization of that formulation.

5. Stochastic approximation EM in the new model

We derive here a SAEM algorithm for the new mixture model (17). Contrary to section 3, we now have three latent random variables $T \sim a(T)$, $t \sim b(t)$, and $\xi \in \mathcal{X}^t$. In analogy to (14), in step 1 of SAEM we need to sample from the posterior

$$p(t, \xi, T \,|\, R; \pi_{k-1}) \propto a(T)\ b(t)\ p(\xi|t;\pi_{k-1})\ p(R|\xi, T)\ . \quad (29)$$

To sample from this posterior we first sample T, then t, then $\xi \in \mathcal{X}^t$. The advantage of sampling in this order is that, since we need to sample only trajectories with nonzero posterior, we can constrain the range of t to $0, .., T$. Further, instead of sampling independent trajectories of different lengths t, we can more efficiently reuse the samples: For given T_i, we generate a single trajectory ξ_i of length T_i and treat all of its sub-trajectories (length-t prefixes of ξ_i) as samples from $p(\xi|t;\pi_{k-1})$, for $t = 0, .., T_i$. We denote the length-t prefix of ξ_i as ξ_{it}. To each of these sub-trajectories ξ_{it} we then assign importance weight

$$w_{\xi_{it}} = b(t)\ p(R{=}1|\xi_{it}, T). \quad (30)$$

The update of $\hat{F}_k$ in step 2 of SAEM reads:

$$\hat{F}_k(\pi) = (1-\alpha_k)\hat{F}_{k\text{-}1}(\pi) + \frac{\alpha_k}{m} \sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} w_{\xi_{it}} \log p(\xi_{it}|t;\pi). \quad (31)$$

Note that this update takes into account all rewards observed during a trajectory ξ_i.

5.1. The finite horizon case

We can also derive a SAEM algorithm for the finite horizon case, in which the value function is the expected sum of rewards up to a terminal time step H:

$$V(\pi) = E\Big[\sum_{t=0}^{H} r_t\ ;\pi\Big]. \quad (32)$$

In the original model of section 2 this value translates to a likelihood, analogous to equation (8), when we choose the time prior $p(T) = 1/(H{+}1)$ for $T \le H$ and zero otherwise. Again, if we derive a SAEM algorithm directly for this original mixture model this leads to an inefficient use of samples. However, we note that the finite horizon case can alternatively be expressed by choosing in equation (17) a delta distribution $a(T) = \delta_{TH}$ and a uniform distribution $b(t) = 1/(H{+}1)$ for $t \le H$, and derive the SAEM algorithm as above. This leads to exactly the same algorithm as above, except that we always choose length $T = H$ to generate the sample trajectories ξ_i. In this case the update of $\hat{F}_k$ reads:

$$\hat{F}_k(\pi) = (1-\alpha_k)\hat{F}_{k\text{-}1}(\pi) + \frac{\alpha_k}{m(H+1)} \sum_{i=1}^{m} \sum_{t=0}^{H} w_{\xi_{it}} \log p(\xi_{it}|t;\pi), \quad (33)$$

where the importance weights now correspond to immediate rewards:

$$w_{\xi_{it}} = p(R{=}1|\xi_{it}, T) = p(R{=}1|u_{|\xi_{it}|}, x_{|\xi_{it}|}). \quad (34)$$

5.2. The tabular case

Here we consider the classical tabular case, where the state and the action spaces are discrete and the policy is parametrized by the conditional probability table π_{ux} of taking action u at state x, with $\sum_u \pi_{ux} = 1$ for all x. For convenience we can write the policy using an indicator function:

$$\pi(u_t|x_t) = \exp \sum_{ux} I(x_t{=}x, u_t{=}u) \log \pi_{ux}. \quad (35)$$

Then the log probability of a sub-trajectory ξ_{it} under the MDP (using (2) and (3), and ignoring terms that do not depend on π) is

$$\log p(\xi_{it}|t;\pi) = \text{const.} + \sum_{ux} c^{ux}_{\xi_{it}} \log \pi_{ux}, \qquad (36)$$

where $c^{ux}_{\xi_{it}}$ is the number of occurrences of a state-action pair (u,x) in the trajectory ξ_{it}, and thereby the number of occurrences of (u,x) in the sample ξ_i until time t. From equation (31) the update of $\hat{F}_k$ reads (ignoring constants):

$$\hat{F}_k(\pi) = (1-\alpha_k)\hat{F}_{k\text{-}1}(\pi) + \frac{\alpha_k}{m} \sum_{ux} (\log \pi_{ux}) \sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} w_{\xi_{it}} c^{ux}_{\xi_{it}}. \qquad (37)$$

In step 3 of the SAEM algorithm we need to find the $\text{argmax}_\pi \hat{F}_k(\pi)$. Let us define another function $Q_k(x,u)$ over all x,u pairs as:

$$Q_k(x,u) = (1-\alpha_k)Q_{k\text{-}1}(x,u) + \frac{\alpha_k}{m} \sum_{i=1}^{m} \frac{1}{|\xi_i|+1} \sum_{t=0}^{|\xi_i|} w_{\xi_{it}} c^{ux}_{\xi_{it}}. \qquad (38)$$

This function can be regarded as an action-value function: since the importance weights $w_{\xi_{it}}$ are proportional to immediate rewards (and they are approximately equal to the immediate rewards when $b(t)$ is nearly constant, e.g., when $\delta \to \gamma$ or in the finite horizon case), the last term in (38) is effectively performing a partial evaluation of the most recently followed policy ($\pi_{k\text{-}1}$) in an approximate batch every-visit Monte Carlo manner. This term is combined with the action value function Q_{k-1} of the previous EM iteration to produce the new Q_k estimate, and so forth for each k. Using the definition of the Q-function, the function $\hat{F}_k(\pi)$ reads

$$\hat{F}_k(\pi) = \sum_{ux} Q_k(x,u) \log \pi_{ux}. \qquad (39)$$

Maximization of $\hat{F}_k(\pi)$ using a Lagrange multiplier $\lambda_x(\sum_u \pi_{ux} - 1)$ for each state x gives[1]

$$\pi_k(x,u) \propto Q_k(x,u). \qquad (40)$$

[1](This footnote does not appear in the proceedings version.) To avoid confusion we clarify that the function $Q_k(x,u)$ defined in eq. (38) is *not* the action-value function of the current policy. More specifically:

- The standard batch every-visit MC estimate of $Q^{\pi_{k-1}}(x,u)$, the action-value function of the current policy π_{k-1}, is equal to the last term in (38) divided by the total number of visits to (x,u) in each trajectory.
- The number of visits to (x,u) depends proportionally on $\pi_{k-1}(x,u)$. Hence, in *expectation* the last term in (38) is proportional to $\pi_{k-1}(x,u)\, Q^{\pi_{k-1}}(x,u)$ (neglecting constants).
- The policy update (40) has a simple interpretation in the case $\alpha = 1$: in *expectation* the update reads $\pi_k(x,u) \propto \pi_{k-1}(x,u)\, Q^{\pi_{k-1}}(x,u)$. (This also explains how the policy can converge to a deterministic behavior in an MDP.)

The smoothness of the policy update operator as a function of the action values has been key in establishing the convergence of related approximate policy iteration algorithms (Jaakkola et al., 1995; Perkins & Precup, 2003).

In summary, SAEM is an optimistic policy iteration algorithm for RL that is similar to Sarsa(1), with a smooth (non-greedy) policy improvement step (40). The algorithm is based on batch every-visit Monte Carlo policy evaluation and involves no bootstrapping, hence it is also appropriate for domains that exhibit partial observability (POMDPs). Its convergence can be guaranteed under quite general conditions (Delyon et al., 1999), but we have not attempted to provide any formal proofs in this paper.

6. Experiments

For illustration purposes we first demonstrate the proposed SAEM algorithm on the 'chain' toy MDP (Dearden et al., 1998). This MDP is shown in Fig. 1. The process always starts from state 1, every chosen action can be flipped with probability 0.2 (hence the state transitions are effectively stochastic), and the discount factor is $\gamma = 0.99$. The optimal policy is to always take action 'a' from any state.

In Fig. 1 we show the learning curve of SAEM (mean and standard deviation of value, over five runs) when initializing with a uniformly random policy. In each iteration of SAEM we sampled $m = 50$ trajectories, and used $\delta = \gamma$, which means that the length of each trajectory was sampled from a geometric distribution with parameter γ, and $b(t)$ is constant. We evaluated each policy by standard MDP policy evaluation using the model. Using the above settings the algorithm was consistently able to locate the optimal policy in all runs. For smaller sample sizes (e.g., $m = 10$) the algorithm would often converge to suboptimal policies corresponding to local maxima of the likelihood function (not shown in the graph).

Since SAEM involves no bootstrapping, it can also be applied on POMDPs. We applied the algorithm on the Hallway POMDP, a standard problem from the

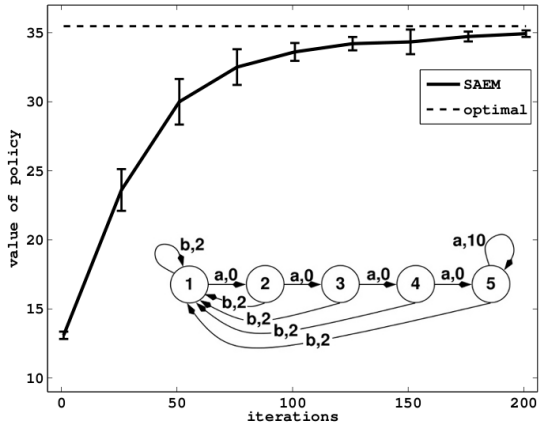

Figure 1. Learning curve of SAEM on the 'chain' MDP. The dashed line corresponds to the optimal value $V^*(1)$.

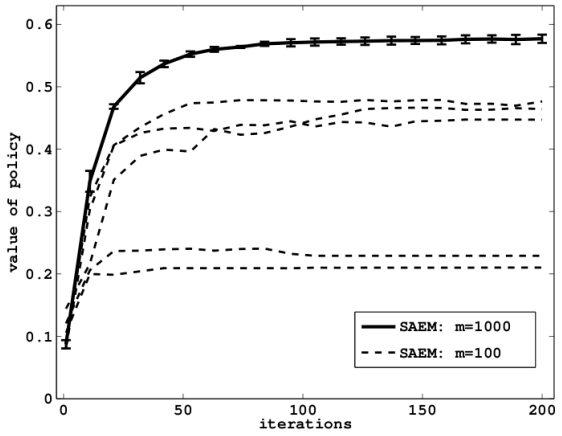

Figure 2. Learning curves of SAEM on the Hallway POMDP, for two different sample sizes.

POMDP literature (Littman et al., 1995). This is a robot maze problem with partial observability, where the robot always starts from an unknown state drawn from a fixed starting distribution and must reach a goal state. This POMDP involves 60 states, 21 observations, and 5 actions. Here $\gamma = 0.95$.

In this experiment we used a stochastic memoryless policy for the agent, which can be regarded as a finite state controller with as many nodes as the number of observations.[2] In Fig. 2 we show the learning curves of SAEM for sample sizes $m = 1000$ and $m = 100$. In both cases we executed five runs, starting with a uniformly random policy, and using $\delta = 0.99$. For $m = 1000$ we plot the mean and standard deviation of the value of each discovered policy (thick line with error bars), and for $m = 100$ we just show the five runs (dashed lines). The value of a policy was computed by standard model-based policy evaluation of finite state controllers (Hansen, 1998).

For small sample sizes ($m = 100$) the algorithm often got trapped in local maxima of the likelihood function (e.g., the lower two dashed lines), but for larger sample sizes ($m = 1000$) the algorithm seemed to be able to avoid bad local maxima, consistently heading for high-value policies (as suggested by the small error bars). This is an encouraging result, confirming the potential of SAEM to handle domains with partial observability.[3]

[2]We chose the class of stochastic memoryless policies just for illustration purposes; it is trivial to adapt SAEM to work with any stochastic finite state controller.

[3]State-of-the-art model-based POMDP solvers produce policies with value around 1 in this problem (Poupart, 2005; Shani et al., 2007).

7. Conclusions

In this paper we reformulated model-free Reinforcement Learning as a probabilistic inference problem (mixture learning). The approach draws on the probabilistic model of Toussaint and Storkey (2006) that allows casting policy optimization as a problem of likelihood maximization in a mixture model. The key to turn this into an efficient model-free RL algorithm was to propose a new mixture model (17) that leads to a stochastic EM algorithm (SAEM) that uses all reward signals on a sampled trajectory. The proposed model can be viewed as a generalization of the stochastic shortest path reformulation of an infinite-horizon MDP (Bertsekas & Tsitsiklis, 1996).

Since the SAEM algorithm is non-bootstrapping it can also be useful in domains exhibiting partial observability. We tried SAEM on a standard problem from the POMDP literature, with encouraging results, providing further evidence that Sarsa-style algorithms hold promise in POMDPs (Loch & Singh, 1998).

The main contribution of this work is the established link between the family of stochastic EM algorithms like SAEM and the family of optimistic policy iteration algorithms like Sarsa(1), which we believe can provide new tools for the design and analysis of RL algorithms.

Acknowledgements

We would like to thank the reviewers for their helpful reviews. Marc Toussaint is supported by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3.

References

Abbeel, P., Coates, A., Quigley, M., & Y., N. A. (2007). An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19*, 1–8. Cambridge, MA: MIT Press.

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

Celeux, G., & Diebolt, J. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Comp. Statis. Quaterly*, *2*, 73–82.

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., & Hinton, G. E. (1997). Using Expectation-Maximization for reinforcement learning. *Neural Computation*, *9*, 271–278.

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. *Proc. 15th National Conf. on Artificial Intelligence* (pp. 761–768). Madison, Wisconsin, USA.

Delyon, B., Lavielle, M., & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, *27*, 94–128.

Gordon, G. (1996). *Chattering in Sarsa(λ)* (Technical Report). CMU Learning Lab internal report.

Hansen, E. (1998). Solving POMDPs by searching in policy space. *Proc. 14th Int. Conf. on Uncertainty in Artificial Intelligence* (pp. 211–219). Madison, Wisconsin, USA.

Hoffman, M., Doucet, A., De Freitas, N., & Jasra, A. (2008). Bayesian policy learning with trans-dimensional MCMC. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*, 665–672. Cambridge, MA: MIT Press.

Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in neural information processing systems 7*, 345–352. MIT Press.

Kober, J., & Peters, J. (2009). Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio and L. Bottou (Eds.), *Advances in neural information processing systems 21*, 849–856.

Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. *Proc. 12th Int. Conf. on Machine Learning* (pp. 362–370).

Loch, J., & Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. *Proc. 15th Int. Conf. on Machine Learning* (pp. 323–331). Madison, Wisconsin, USA.

Melo, F. S., Meyn, S. P., & Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. *Proc. 25th Int. Conf. on Machine Learning* (pp. 664–671). Helsinki, Finland.

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

Perkins, T. J., & Pendrith, M. D. (2002). On the existence of fixed points for Q-learning and Sarsa in partially observable domains. *Proc. 19th Int. Conf. on Machine Learning* (pp. 490–497).

Perkins, T. J., & Precup, D. (2003). A convergent form of approximate policy iteration. In S. T. S. Becker and K. Obermayer (Eds.), *Advances in neural information processing systems 15*, 1595–1602. Cambridge, MA: MIT Press.

Peters, J., & Schaal, S. (2008). Learning to control in operational space. *International Journal of Robotics Research*, *27*, 197–212.

Poupart, P. (2005). *Exploiting structure to efficiently solve large scale partially observable Markov decision processes.* Doctoral dissertation, Dept. of Computer Science, University of Toronto.

Shani, G., Brafman, R. I., & Shimony, S. E. (2007). Forward search value iteration for POMDPs. *In Int. Joint Conf. on Artificial Intelligence* (pp. 2619–2624).

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

Toussaint, M., & Storkey, A. (2006). Probabilistic inference for solving discrete and continuous state Markov decision processes. *Proc. 23rd Int. Conf. on Machine Learning* (pp. 945–952). Pittsburgh, Pennsylvania, USA.

Tsitsiklis, J. N. (2002). On the convergence of optimistic policy iteration. *Journal of Machine Learning Research*, *3*, 59–72.

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