Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Online Expectation Maximization for Reinforcement Learning in POMDPs

Miao Liu, Xuejun Liao, Lawrence Carin
{miao.liu, xjliao, lcarin}@duke.edu
Duke University,
Durham, NC 27708, USA

Abstract

We present online nested expectation maximization for model-free reinforcement learning in a POMDP. The algorithm evaluates the policy only in the current learning episode, discarding the episode after the evaluation and memorizing the sufficient statistic, from which the policy is computed in closed-form. As a result, the online algorithm has a time complexity $O(n)$ and a memory complexity $O(1)$, compared to $O(n^2)$ and $O(n)$ for the corresponding batch-mode algorithm, where n is the number of learning episodes. The online algorithm, which has a provable convergence, is demonstrated on five benchmark POMDP problems.

1 Introduction

The policy of a partially observable Markov decision process (POMDP) is a mapping from belief-states to actions [Kaelbling *et al.*, 1998]. For a given POMDP, the optimal policy can be obtained by solving a Markov decision process in belief-states. In reinforcement learning (RL), however, the agent must learn a policy via experiencing with the POMDP, without assuming knowledge of the process's true underlying model. In this case, the policy can be obtained by using either of the following approaches: (i) a model-based approach [Poupart and Vlassis, 2008; Doshi-Velez *et al.*, 2009; Doshi-Velez, 2010], in which the agent first learns the underlying model using its experience with the POMDP and then solves the learned model to obtain an approximate policy; (ii) a model-free approach [Li *et al.*, 2009; Cai *et al.*, 2009; Liu *et al.*, 2011], in which the agent learns the policy directly from the experience, skipping the model-learning step.

The two approaches each have their advantages and disadvantages. Since the underlying model is a complete description of the POMDP, a learned model can be used not only to find the optimal policy but for other purposes also; thus, a model-based approach is versatile. However, the agent needs a comprehensive set of experiences with the POMDP to make the learned model complete, implying the model-learning step is not efficient in time. In addition, the agent needs a second step to solve the learned model to the get the policy. Although modern POMDP solvers can find approximate policies efficiently, the efficiency can be plagued when the solvers must be invoked repeatedly, as in sample-based methods [Doshi-Velez *et al.*, 2009; Doshi-Velez, 2010].

A model-free approach, by contrast, focuses exclusively on the policy. Though one can find the optimal policy given the model, inverting the learned policy into the model is generally not possible. The fact that the policy is not as rich in information as the model implies: (i) learning a policy may need only a subset of the experience required in learning a complete underlying model of the POMDP; (ii) the policy can only be used for a subset of the purposes which the model can be used for. For these reasons, a model-free approach is typically more efficient in time, but less versatile, than its model-based counterpart; moreover, the avoidance of a second step of model-to-policy conversion enhances the time efficiency.

The choice between a model-based approach and a model-free approach is, therefore, a tradeoff between versatility and time efficiency. When the policy is the primary interest, the right choice would be an approach that is just versatile enough for finding the optimal policy, but not more. This idea has been pursued in model-based methods. For example, the work in [Wierstra and Wiering, 2004] approximates a POMDP as an utile distinction hidden Markov model which creates memory to preserve perceptual and utility distinction only when necessary. The recent work in [Grady *et al.*, 2013] solves a less complex model, with reduced state and action spaces, to find approximate policies for the original POMDP. Both these methods solve a compressed model of the original POMDP to find approximate policies, enhancing time efficiency while reducing the model's versatility. A model-free approach can be considered as an extreme case, in which the POMDP is compressed into the policy itself, i.e., the most compressed "model" that preserves the policy.

Most model-free methods for RL in a POMDP are based on representing the policy as a finite state controller (FSC). Early work learns FSCs using stochastic gradient descent [Meuleau *et al.*, 1999; Aberdeen and Baxter, 2002]. Recently, it has been shown in [Li *et al.*, 2009] that expectation maximization (EM), a popular algorithm in statistics [Dempster *et al.*, 1977], provides an effective tool for learning a family of policies that include FSCs as a special case. The policies considered in [Li *et al.*, 2009], collectively referred to as *regionalized policy representation (RPR)*, treat belief-state as a latent random variable and integrate it out to yield a marginalized policy that is expressed as a probability distribution of the

current action conditional on the history of past actions and observations. The RPR is a general form of FSC, where each internal memory unit (called a decision state in the RPR and a machine node in the FSC) is associated with a distribution of actions, instead of a single action; as a result, the transition from unit z to unit z' depends jointly on the action at z and the observation at z', instead of on the observation only. The action-dependency reflects the inherent uncertainty of the action choice in each belief region, which, as discussed in depth in [Sondik, 1978], cannot be resolved unless the policy is finite transient, in which case the RPR specializes to a FSC.

The optimal RPR can be learned by nested expectation maximization [Li *et al.*, 2009], where the outer-loop EM evaluates the policy in the E-step and improves the policy in the M-step, and the inner-loop EM realizes the outer-loop M-step. It is noteworthy that the nested EM maximizes a value function, which cannot be interpreted as a probability of the observed variables as in the case of a likelihood function.

The nested EM is better interpreted as a consequence of the policy improvement theorem of a POMDP [Blackwell, 1965]. As detailed in [Li *et al.*, 2009, pages 1140-1143], the outer-loop E-step re-computes the immediate rewards such that a discounted sum of the new rewards, averaged over the episodes, represents the value of the current policy; in addition, the inner-loop EM updates the policy by maximizing the weighted log-likelihood function of a cost-sensitive hidden Markov model, with the weights constituted by the re-computed new rewards. The weighted log-likelihood function corresponds to the logarithm of the action-value function in [Blackwell, 1965], parameterized by the RPR and averaged over all belief points, and the average value is maximized by the inner-loop EM to yield an improved policy.

The RPR has been introduced in [Li *et al.*, 2009] as a tool for multi-task RL across multiple POMDPs. Subsequent work has focused on using the RPR for reinforcement learning in a single POMDP [Cai *et al.*, 2009; Liu *et al.*, 2011]. All previous work assumes batch-mode learning, updating the policy based on the entire set of experiences collected so far. Consequently, as new experiences arrive sequentially, the computational cost and memory demand increase fast.

In this paper, we aim to reduce the time and memory complexity of learning an RPR by using online nested expectation maximization. The major change from the batch-mode EM is that the policy is updated each time based on a partial policy evaluation, with only the latest episode of agent-environment interaction used in the outer-loop E-step and inner-loop E-step. Therefore each episode's contribution to the sufficient statistics is computed only when the episode is new, instead of computed repeatedly when each subsequent episode comes in; this makes the E-steps computationally more efficient. Furthermore, as each episode is used only when it is new, the algorithm need only memorize the latest episode, discarding all older ones, leading to tremendous memory savings.

2 Regionalized Policy Representation

Definition 1. [Li *et al.*, 2009] A regionalized policy representation is a tuple $(\mathcal{A}, \mathcal{O}, \mathcal{Z}, W, \mu, \pi)$, where $\mathcal{A}$, $\mathcal{O}$, and $\mathcal{Z}$ are respectively a finite set of actions, observations, and decision states; W is a set of Markov transition matrices, with $W^{zao}_{z'}$ denoting the probability of transiting from decision state z to z' when action a in z results in observation o in z'; μ is the initial distribution of decision states, with μ_z the probability of initially being in z; π is a set of stochastic policies, with π^z_a the probability of taking action a in z.

For simplicity, $\mathcal{Z}$ is denoted as $\{1, 2, \cdots, |\mathcal{Z}|\}$, where $|\mathcal{Z}|$ is the cardinality; similar notations are used for $\mathcal{A}$ and $\mathcal{O}$. The set of RPR parameters is denoted as $\Theta = \{\pi, \mu, W\}$. A consecutively indexed variable is abbreviated as the variable with its index range, which is indicated in the subscript of the variable; for example, $a_{0:T} = (a_0, a_1, \ldots, a_T)$, $W^{zao}_{1:|\mathcal{Z}|} = (W^{zao}_1, W^{zao}_2, \cdots, W^{zao}_{|\mathcal{Z}|})$, etc.

Given $h_t = \{a_{0:t-1}, o_{1:t}\}$, the history of actions and observations up to t, the RPR chooses action a_t according to

$$p(a_t|h_t, \Theta) = \frac{p(a_{0:t}|o_{1:t}, \Theta)}{p(a_{0:t-1}|o_{1:t}, \Theta)} = \frac{p(a_{0:t}|o_{1:t}, \Theta)}{p(a_{0:t-1}|o_{1:t-1}, \Theta)}, \quad (1)$$

where the second equality arises because o_t has no influence on the actions before t, and $p(a_{0:t}|o_{1:t}, \Theta)$ is a marginal of

$$p(a_{0:t}, z_{0:t}|o_{1:t}, \Theta) = \mu_{z_0} \pi^{z_0}_{a_0} \prod_{\tau=1}^{t} W^{z_{\tau-1} a_{\tau-1} o_\tau}_{z_\tau} \pi^{z_\tau}_{a_\tau}, \quad (2)$$

by integrating out latent decision states $z_{0:t}$. It follows from (1) that $p(a_{0:t}|o_{1:t}, \Theta) = \prod_{\tau=0}^{t} p(a_\tau|h_\tau, \Theta)$.

The RPR parameters are learned from the agent experiences by using an empirical value function defined below. Assuming the agent-POMDP interaction is episodic [Sutton and Barto, 1998], the experiences are represented as a set of episodes. An episode of length T_k is denoted by $(a^k_0 r^k_0 o^k_1 a^k_1 r^k_1 \cdots o^k_{T_k} a^k_{T_k} r^k_{T_k})$, where r is a nonnegative immediate reward, k indexes the episodes, and the subscripts index discrete time steps.

Definition 2. [Li *et al.*, 2009] Let $\mathcal{D}^{(K)} = \{(a^k_0 r^k_0 o^k_1 a^k_1 r^k_1 \cdots o^k_{T_k} a^k_{T_k} r^k_{T_k})\}_{k=1}^{K}$ be a set of episodes resulting from the interaction between the POMDP and an agent who chooses actions according to Π, an arbitrary stochastic policy with action-selecting distributions $p^\Pi(a|h) > 0$, $\forall$ action a, $\forall$ history h. The *empirical value function* is defined as

$$\hat{V}(\mathcal{D}^{(K)}; \Theta) \stackrel{def.}{=} \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T_k} \gamma^t r^k_t \frac{\prod_{\tau=0}^{t} p(a^k_\tau|h^k_\tau, \Theta)}{\prod_{\tau=0}^{t} p^\Pi(a^k_\tau|h^k_\tau)} \quad (3)$$

where $h^k_t = (a^k_{0:t-1}, o^k_{1:t})$, $0 < \gamma < 1$ is the discount as defined in the POMDP.

It has been shown in [Li *et al.*, 2009] that $\lim_{K\to\infty} \hat{V}(\mathcal{D}^{(K)}; \Theta)$ is the expected sum of discounted rewards by following the RPR parameterized by Θ for an infinite number of steps. Therefore, the RPR resulting from maximization of the empirical value function is an approximation of the optimal policy, assuming the number of decision states, i.e., $|\mathcal{Z}|$, is large enough to accommodate the optimal policy. The optimal policy of a POMDP can be represented by a RPR because the RPR subsumes as a special case the finite state controller (FSC) [Li *et al.*, 2009], which is known to approximate the optimal policy of any POMDP

to a arbitrary precision [Sondik, 1978]. The optimal $|\mathcal{Z}|$ can be inferred by the method in [Liu *et al.*, 2011].

The batch-mode nested EM algorithm in [Li *et al.*, 2009] is re-stated in Theorem 3.

Theorem 3. *Let $\{\Theta_n\}_{n\geq 0}$ be a sequence produced by the iterative update $\Theta_{n+1} = \arg\max_{\hat{\Theta}\in\Xi} \mathrm{LB}(\hat{\Theta}|\Theta_n)$, where Θ_0 is an arbitrary initialization and*

$$\Xi = \Big\{\Theta = (\mu, \pi, W):\ \textstyle\sum_{j=1}^{|\mathcal{Z}|}\mu_j = 1, \sum_{a=1}^{|\mathcal{A}|}\pi_a^i = 1, \sum_{j=1}^{|\mathcal{Z}|} W_j^{iao} = 1, i = 1:|\mathcal{Z}|, a = 1:|\mathcal{A}|,\ o = 1:|\mathcal{O}|\Big\},$$

$$\mathrm{LB}(\hat{\Theta}|\Theta_n) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T_k}\gamma^t\nu_t^k\Bigg\{\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,0}^k(i)\ln\hat{\mu}_i + \sum_{\tau=0}^{t}\Bigg[\sum_{i=1}^{|\mathcal{Z}|}\phi_{t,\tau}^k(i)\ln\hat{\pi}_{a_\tau^k}^i + \sum_{i,j=1}^{|\mathcal{Z}|}\xi_{t,\tau}^k(i,j)\ln\hat{W}_i^{j,a_{\tau-1}^k,o_\tau^k}\Bigg]\Bigg\},$$

$$\nu_t^k = \frac{r_t^k p(a_{0:t}^k|o_{1:t}^k,\Theta_n)}{\prod_{\tau=0}^t p^{\Pi}(a_\tau^k|h_\tau^k)}, \forall\, t,k, \tag{4}$$

$$\xi_{t,\tau}^k(i,j) = p(z_\tau^k = i, z_{\tau+1}^k = j|a_{0:t}^k, o_{1:t}^k, \Theta_n), \tag{5}$$

$$\phi_{t,\tau}^k(i) = p(z_\tau^k = i|a_{0:t}^k, o_{1:t}^k, \Theta_n), \tag{6}$$

then $\{\Theta_n\}_{n\geq 0}$ monotonically increases (3), until convergence to a maxima.

The quantity ν_t^k in (4) is the re-computed reward at time t in episode k; it represents the expected reward of following policy Θ_n, and is computed in the outer-loop E-step. The quantities in (5)-(6) are standard soft counts resulting from the Baum-Welch formula for learning an HMM [Rabiner, 1989], and they are computed in the inner-loop E-step. Therefore, (4) is computed less frequently than (5)-(6); that is, once (4) is computed, it is not computed again until the inner-loop EM converges to the desired precision. In practice, one may compute (4) after running the inner-loop EM only a few iterations, instead of waiting for its convergence; we find this works well in our experiments.

3 Online Nested Expectation Maximization

To prepare for the online EM algorithm, we rewrite the local objective in Theorem 3 equivalently as

$$\mathrm{LB}(\hat{\Theta}|\Theta_n) = \frac{1}{K}\sum_{i=1}^{|\mathcal{Z}|}\upsilon_i\ln\hat{\mu}_i + \frac{1}{K}\sum_{i=1}^{|\mathcal{Z}|}\sum_{a=1}^{|\mathcal{A}|}\rho_a^i\ln\hat{\pi}_a^i + \frac{1}{K}\sum_{a=1}^{|\mathcal{A}|}\sum_{o=1}^{|\mathcal{O}|}\sum_{i=1}^{|\mathcal{Z}|}\sum_{j=1}^{|\mathcal{Z}|}\omega_j^{iao}\ln\hat{W}_j^{iao}, \tag{7}$$

where, for any $i,j\in\mathcal{Z}$, $a\in\mathcal{A}$, and $o\in\mathcal{O}$,

$$\upsilon_i = \tfrac{1}{K}\textstyle\sum_{k=1}^K\sum_{t=0}^{T_k}\gamma^t\nu_t^k\phi_{t,0}^k(i), \tag{8}$$

$$\rho_a^i = \tfrac{1}{K}\textstyle\sum_{k=1}^K\sum_{t=0}^{T_k}\gamma^t\nu_t^k\sum_{\tau=0}^t\phi_{t,\tau}^k(i)\,\delta(a_\tau^k, a), \tag{9}$$

$$\omega_j^{iao} = \tfrac{1}{K}\textstyle\sum_{k=1}^K\sum_{t=0}^{T_k}\gamma^t\nu_t^k\sum_{\tau=1}^{t-1}\xi_{t,\tau}^k(i,j)\delta(a_\tau^k,a)\delta(o_{\tau+1}^k,o), \tag{10}$$

and $\delta(a,b)$ is one if $a=b$ and zero otherwise. The problem $\max_{\hat{\Theta}\in\Xi}\mathrm{LB}(\hat{\Theta}|\Theta_n)$ has a unique solution given by

$$\hat{\mu}_i = \frac{\upsilon_i}{\sum_{i=1}^{|\mathcal{Z}|}\upsilon_i},\ \hat{\pi}_a^i = \frac{\rho_a^i}{\sum_{a=1}^{|\mathcal{A}|}\rho_a^i},\ \hat{W}_j^{iao} = \frac{\omega_j^{iao}}{\sum_{j=1}^{|\mathcal{Z}|}\omega_j^{iao}}, \tag{11}$$

for $i,j\in\mathcal{Z}$, $a\in\mathcal{A}$, and $o\in\mathcal{O}$. Define

$$f(\hat{\Theta};\upsilon,\rho,\omega) \overset{Def.}{=} \mathrm{LB}(\hat{\Theta}|\Theta_n) + (1-\textstyle\sum_{i=1}^{|\mathcal{Z}|}\hat{\mu}_i)\sum_{i=1}^{|\mathcal{Z}|}\upsilon_i + \sum_{a=1}^{|\mathcal{A}|}\sum_{o=1}^{|\mathcal{O}|}\sum_{i=1}^{|\mathcal{Z}|}(1-\sum_{j=1}^{|\mathcal{Z}|}\hat{W}_j^{iao})\sum_{j=1}^{|\mathcal{Z}|}\omega_j^{iao} + \sum_{i=1}^{|\mathcal{Z}|}(1-\sum_{a=1}^{|\mathcal{A}|}\hat{\pi}_a^i)\sum_{a=1}^{|\mathcal{A}|}\omega_a^i. \tag{12}$$

One may readily verify that $f(\hat{\Theta};\upsilon,\rho,\omega)$ has a unique stationary point that is equal to $\hat{\Theta}$ as given in (11).

Let $\boldsymbol{\theta} = [\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{w}]^T$ and $\boldsymbol{s} = [\boldsymbol{\upsilon},\boldsymbol{\rho},\boldsymbol{\omega}]^T$, with the definitions $\boldsymbol{\mu} = [\mu_i]_{i=1:|\mathcal{Z}|}$, $\boldsymbol{\pi} = [\pi_a^i]_{a=1:|\mathcal{A}|,i=1:|\mathcal{Z}|}$, $\boldsymbol{w} = [W_j^{iao}]_{j=1:|\mathcal{Z}|,i=1:|\mathcal{Z}|,a=1:|\mathcal{A}|,o=1:|\mathcal{O}|}$, and similarly $\boldsymbol{\upsilon} = [\upsilon_i]_{i=1:|\mathcal{Z}|}$, $\boldsymbol{\rho} = [\rho_a^i]_{a=1:|\mathcal{A}|,i=1:|\mathcal{Z}|}$, $\boldsymbol{\omega} = [\omega_j^{iao}]_{j=1:|\mathcal{Z}|,i=1:|\mathcal{Z}|,a=1:|\mathcal{A}|,o=1:|\mathcal{O}|}$, where the elements in a vectorization are arranged by following a left-to-right order in sorting the indices. For example, $[\pi_a^i]_{a=1:|\mathcal{A}|,i=1:|\mathcal{Z}|} = [\pi_1^1,\cdots,\pi_{|\mathcal{A}|}^1,\cdots,\pi_1^{|\mathcal{Z}|},\cdots,\pi_{|\mathcal{A}|}^{|\mathcal{Z}|}]$.

Define $\boldsymbol{C} = \mathrm{diag}(\boldsymbol{C}_\mu,\boldsymbol{C}_\pi,\boldsymbol{C}_W)$, where $\boldsymbol{C}_\mu$ is a $|\mathcal{Z}|\times|\mathcal{Z}|$ matrix of ones, $\boldsymbol{C}_\pi = \mathrm{diag}\{(\boldsymbol{C}_\pi^i)_{i=1:|\mathcal{Z}|}\}$ with $\boldsymbol{C}_\pi^i$ a $|\mathcal{A}|\times|\mathcal{A}|$ matrix of ones, and $\boldsymbol{C}_W = \mathrm{diag}\{(\boldsymbol{C}_W^{iao})_{i=1:|\mathcal{Z}|,a=1:|\mathcal{A}|,o=1:|\mathcal{O}|}\}$ with $\boldsymbol{C}_W^{iao}$ a $|\mathcal{Z}|\times|\mathcal{Z}|$ matrix of ones. By construction, $\boldsymbol{C}$ is a block-diagonal matrix with each block a matrix of all ones, and each block is associated with the corresponding sub-vector in $\boldsymbol{\theta}$ whose elements sum up to one.

Let $\ln(\boldsymbol{\theta})$ denote a vector resulting from element-wise application of the natural logarithm to $\boldsymbol{\theta}$. Define

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \ln\boldsymbol{\theta} - \boldsymbol{C}\boldsymbol{\theta}. \tag{13}$$

Then one can rewrite (12) as

$$f(\hat{\boldsymbol{\theta}};\boldsymbol{s}) = \boldsymbol{s}^T\boldsymbol{\psi}(\hat{\boldsymbol{\theta}}), \tag{14}$$

and the solution in (11) is a solution to $\nabla_{\hat{\boldsymbol{\theta}}}f(\hat{\boldsymbol{\theta}};\boldsymbol{s}) = \boldsymbol{0}$ or equivalently $\nabla_{\hat{\boldsymbol{\theta}}}\boldsymbol{\psi}^T(\hat{\boldsymbol{\theta}})\boldsymbol{s} = \boldsymbol{0}$, where $\nabla_{\boldsymbol{\theta}}$ denotes the gradient with respect to $\boldsymbol{\theta}$.

Using the vector notation, one can restate the algorithm in Theorem 3 as:

Outer- and/or inner- loop E-steps: Update the sufficient statistic $\boldsymbol{s} = \boldsymbol{s}(\mathcal{D}^{(K)},\boldsymbol{\theta}_{n-1})$ using (8)-(10) and (4)-(6).

Inner-loop M-step: Solve $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T(\boldsymbol{\theta})\boldsymbol{s}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_n} = \boldsymbol{0}$ for $\boldsymbol{\theta}_n$.

We have written $\boldsymbol{s} = \boldsymbol{s}(\mathcal{D}^{(K)},\boldsymbol{\theta}_{n-1})$ to emphasize that the sufficient statistic is a function of the previous parameter vector $\boldsymbol{\theta}_{n-1}$ and episodes $\mathcal{D}^{(K)}$. Note that the constraint $\hat{\Theta}\in\Xi$ in Theorem 3 has been accounted for in (12) and (14).

To see the possibility of an online version of the EM algorithm, we note from (8)-(10) that the sufficient statistic is a sum over the episodes, and therefore one can write

$$\boldsymbol{s}(\mathcal{D}^{(n)},\boldsymbol{\theta}_{n-1}) = \tfrac{1}{n}\textstyle\sum_{k=1}^n\boldsymbol{s}(\varepsilon_k,\boldsymbol{\theta}_{n-1}), \tag{15}$$

where $\boldsymbol{s}(\varepsilon_k,\boldsymbol{\theta}_{n-1})$, a vector collecting only the terms with index k in (8)-(10), represents the contribution from ε_k, and $\mathcal{D}^{(n)} = \{\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_n\}$ are episodes coming in sequentially.

The online EM algorithm employs a noisy (indicated by the hat above $\boldsymbol{s}$) version of the true sufficient statistic in (15),

$$\hat{\boldsymbol{s}}(\mathcal{D}^{(n)},\boldsymbol{\theta}_{n-1}) = \textstyle\sum_{k=1}^n\beta_k\boldsymbol{s}(\varepsilon_k,\boldsymbol{\theta}_{k-1}), \tag{16}$$

where $\beta_k = \alpha_k \prod_{j=k+1}^{n}(1-\alpha_j)$, and $\{0 < \alpha_k < 1\}_{k=1}^{n}$, referred to as learning rates, discount the contribution from each episode, with earlier episodes giving greater discounts because they are computed using earlier versions of $\boldsymbol{\theta}$ and are thus less accurate. One can equivalently write (16) as

$$\hat{\boldsymbol{s}}(\mathcal{D}^{(n)}, \boldsymbol{\theta}_{n-1}) = \hat{\boldsymbol{s}}_n, \tag{17}$$

with the recursive definition

$$\hat{\boldsymbol{s}}_k = (1-\alpha_k)\hat{\boldsymbol{s}}_{k-1} + \alpha_k \boldsymbol{s}(\varepsilon_k, \boldsymbol{\theta}_{k-1}), \quad k = 1, \cdots, n. \tag{18}$$

The online nested EM algorithm processes the episodes on the fly, and removes an episode from the memory once its contribution to the sufficient statistic has been recorded using (18). The contribution from the k-th episode, $\boldsymbol{s}(\varepsilon_k, \boldsymbol{\theta}_{k-1})$, is computed using (8)-(10) and (4)-(6), considering only the terms with index k. Recalling that (4) constitutes the outer-loop E-step and (5)-(6) the inner-loop E-step, one need only compute (4) every δn iterations of the inner-loop EM, where δn may be determined by the convergence of the inner-loop EM. When n is small, however, the sufficient statistics is very noisy, and in this case it is preferable to run only a few iterations for both the outer-loop EM and the inner-loop EM, to prevent premature convergence. The RPR parameter vector is updated in the inner-loop M-step, as the solution to

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}^T(\boldsymbol{\theta}) \hat{\boldsymbol{s}}_n = \mathbf{0}. \tag{19}$$

The solution, given in closed-form in (11), is essentially a set of normalized sub-vectors of the sufficient statistic.

The complete online algorithm is given in Algorithm 3. The algorithm performs on-policy learning [Sutton and Barto, 1998], where the agent learns the RPR while also using it to choose actions in collecting experiences. In order to achieve a balance between exploration and exploitation, the behavior policy used in the algorithm, Π_n, is a mixture of two experts [Jordan and Jacobs, 1994], i.e.,

$$p^{\Pi_n}(a|h) = p(y=0|h)p(a|h,\Theta_n) + p(y=1|h)/|\mathcal{A}|, \ \forall\, h,$$

where the first expert, $p(a|h,\Theta_n)$, is the RPR policy as given by (1), the second is a random policy, and the gating network $p(y|h)$ is learned, along with the EM algorithm, using the dual-policy approach described in [Cai *et al.*, 2009]. The behavior policy used here generalizes the ϵ-greedy policy [Sutton and Barto, 1998] in such a sense that the probability of choosing random actions is not a constant ϵ; rather, it depends on the history of past actions and observations.

3.1 Time and memory complexity

The batch-mode EM algorithm has a time complexity of $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{m=1}^{n}\sum_{k=1}^{m} T_k^2\big)$ when there is a nonzero reward at every step in an episode (dense reward), or $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{m=1}^{n}\sum_{k=1}^{m} T_k\big)$ when nonzero reward is received only at the terminal step in each episode.

In comparison, the proposed online EM algorithm has a time complexity of $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{k=1}^{n} T_k^2\big)$ in the densely-rewarded case, and $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 \sum_{k=1}^{n} T_k\big)$ in the terminally-rewarded case.

When all episodes have the same number of steps, say T, the complexity of batch-mode EM becomes $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T^2 n^2\big)$ in the case of dense rewards and $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T n^2\big)$ in the case of terminal rewards, while the complexity of online EM is $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T^2 n\big)$ and $O\big(|\mathcal{A}||\mathcal{O}||\mathcal{Z}|^2 T n\big)$, in the two respective cases.

Therefore, the accumulative learning time, as a function of the number of episodes n, is approximately on the order of $O\big(n^2\big)$ for batch-mode EM, while it is approximately on the order of $O\big(n\big)$ for online EM.

The batch algorithm stores all episodes, and thus it has a memory complexity of $O(n)$. The online algorithm stores only the latest episode; its memory complexity is $O(1)$.

Algorithm 1 The online nested EM algorithm for RPR

Initialize $n = 0$, $\alpha_0 \ll 1$, $\hat{\mathbf{s}}_0 = \mathbf{0}$, Π_0 as a random policy,
while $P_{\text{explore}} = \mathbb{E}[p(y=1|h)] < \epsilon$ **do**
1. collect the n-th episode ε_n following Π_{n-1}
2. (outer-loop E-step) re-compute rewards with (4).
while inner-loop EM not convergent **do**
1. (inner-loop E-step) compute (5)-(6).
2. update the sufficient statistic using (18).
3. (inner-loop M-step) update the RPR with (11).
end while
3. (exploitation-exploration balance) update Π_n (the gating network) using the method in [Cai *et al.*, 2009].
4. adjust learning rate α_n
5. increase pointer $n := n + 1$.
end while

3.2 Convergence Analysis

The convergence of the online algorithm depends on the asymptotic behavior of the noisy sufficient statistic in (18). If $\hat{\boldsymbol{s}}_n$ approaches to the accurate (noise-free) sufficient statistic $\boldsymbol{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta}_n)$ as n is sufficiently large, the online algorithm converges to the batch-mode solution after n iterations; thus, the online algorithms also converges to the optimal RPR, considering that the batch-mode solution gives the optimal RPR when the number of episodes is sufficiently large (see the discussion below Definition 2).

Theorem 4 shows that the noise in $\hat{\boldsymbol{s}}_n$ indeed vanishes as $n \to \infty$, under certain conditions on the learning rates $\{\alpha_n\}$. Thus convergence of the online algorithm is guaranteed.

Theorem 4. *If* $\sum_{n=1}^{\infty} \alpha_n = \infty$ *and* $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$, *then* $\lim_{n\to\infty} \|\boldsymbol{s}(\mathcal{D}^{(\infty)}, \boldsymbol{\theta}_n) - \hat{\boldsymbol{s}}_n\| = 0$.

Theorem 4 can be proven using the techniques in [Delyon *et al.*, 1999] and [Cappé and Moulines, 2009]. The details are omitted here to conserve space.

4 Experiments

We investigate the empirical performance of the online algorithm on the five benchmark problems described in Table 1. The POMDP models for these problems are available at http://www.cs.brown.edu/research/ai/pomdp/examples. These models are assumed unknown to the agent; they are used as black boxes to simulate the episodes of agent-environment interaction. For all problems, an episode starts from a random position and ends when the goal is achieved.

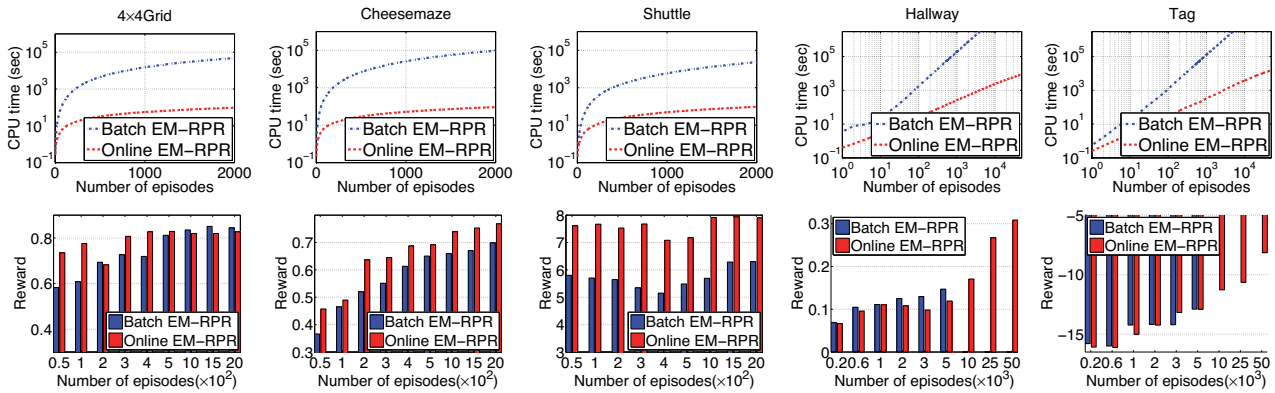

Figure 1: Performance Comparison between online and batch EM-RPR learning algorithms. Top row and bottom correspond to averaged accumulative CPU time and reward respectively. The columns from left to right correspond to 5 benchmark problems; for all of the latter, the horizontal axis corresponds to the number of episodes seen by the agent when developing the policy. The results of batch EM for $n > 2000$ are still being computed at the time of submission and will be provided in the final version of the paper.

Table 1: Five benchmark POMDP problems

Name	$\|\mathcal{A}\|$	$\|\mathcal{O}\|$	Number of States
Shuttle	8	3	3
Cheese maze	4	7	11
4×4 grid	4	2	16
Hallway	5	21	60
Tag	5	30	870

4.1 Online RPR versus Batch-mode RPR

We compare the performance of the online RPR against that of the batch-mode RPR. The performance is measured by the discounted accumulative reward averaged over 5 independent runs; in each run, we use 101 (or 251) test episodes of at most 101 (or 251) steps for the first three (or other) problems. The computational efficiency is compared based on the accumulative CPU time in the learning phase. The number of states is set to $|\mathcal{Z}| = 20$ for the first three small problems, $|\mathcal{Z}| = 40$ for Hallway and Tag. The comparison results are plotted in Figure 1. The performance for the online-learned policy as a function of $|\mathcal{Z}|$ (not provided here due to the page limit) shows that the policy is robust over a wide range of $|\mathcal{Z}|$. Automatic inference of $|\mathcal{Z}|$ can be performed using the infinite-RPR framework in [Liu *et al.*, 2011].

As can be seen from the leftmost three columns of Figure 1, the online EM algorithm shows some advantages in performance, especially during the early stage of learning when episodes are collected by exploration. A comparison of CPU time shows that online EM is significantly more efficient than the batch EM, and the time agrees with the complexity analysis in Section 3.1. Moreover it is noted that, in the Shuttle and Cheesemaze problems, batch EM is prone to local optima, whereas online EM is a stochastic algorithm which seems capable of jumping out of local optima to obtain better solutions.

The rightmost two columns in Figure 1 demonstrate that the online algorithm makes the RPR scale to larger problems, and achieves a better tradeoff between the number of episodes (n) and the policy quality. When n is small, batch EM is better than online EM, which can be explained by the fact that the batch method updates the sufficient statistics using all available data. However, given the same amount of learning time, the online RPR achieves better policy performance; for example, by the time that online EM has processed $50,000$ episodes and achieved a test reward of 0.3 for Hallway and -8 for Tag, batch EM has processed less than 1000 episodes, achieving only 0.1 and -14 for these two problems, respectively. These comparisons show the RPR can scale up to larger POMDP problems with online learning.

4.2 Comparison with Other Methods

We compare the RPR policy learned with the online algorithm to three existing algorithms, which all assumes the POMDP models are unknown. The first one is McCallum's U-tree [McCallum, 1995], including its modified version [Zheng and Cho, 2011], which, like the RPR, is a policy-based approach and does not attempt to learn the POMDP models. The second is iPOMDP [Doshi-Velez *et al.*, 2009], including its variant Policy Prior (PP) [Doshi-Velez, 2010], which learn both POMDP models and control policies. The last competing algorithm is mixture learning [Vlassis and Toussaint, 2009], referred to MDP-EM, which is an online EM algorithm treating the observation in a POMDP as the state in an MDP. We do not repeat the experiments for the three competing methods, but rather cite the results from the literature. Since the cited results are based on different testing procedures, we report the comparison to each method separately, using the same test procedure as in the literature for each respective method.

We compare to the U-tree results as reported in [Zheng and Cho, 2011]. To make the results comparable, we strictly follow the same procedures to set up our experiment for the RPR. Specifically, the experiment for each problem consists of 21 independent trials. In each trial, we consider three settings of $|\mathcal{Z}|$ for the RPR policy, i.e., $|\mathcal{Z}| \in \{10, 20, 30\}$; for each setting, the RPR policy is learned using the online EM algorithm, based on episodes collected by following the behavior policy for 75000 steps, and then the learned RPR policy is followed 25000 steps to test the its quality. The quality in each trial is measured by the average reward earned over

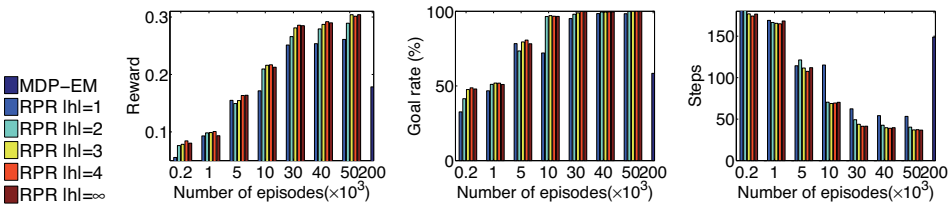

Figure 2: A comparison of online EM-RPR to EM-MDP on Hallway. $|h|$ is the number of observations. The performance of the EM-MDP is reported only for $n = 200 \times 10^3$.

Table 2: A comparison, in terms of averaged reward, to U-trees on small problems. The parenthesized integers indicate the numbers of nodes used in U-trees, which correspond to the numbers of decision states $|\mathcal{Z}|$ for the RPR.

Methods	Shuttle	Cheese Maze	4×4 Grid
U-Tree	1.833 (62)	0.184 (53)	0.179 (47)
Modified U-Tree	1.820 (25)	0.184 (19)	0.179 (20)
RPR ($\|\mathcal{Z}\| = 10$)	1.76± 0.136	0.161± 0.018	0.259±0.033
RPR ($\|\mathcal{Z}\| = 20$)	1.824 ± 0.046	0.151 ± 0.021	0.207± 0.049
RPR ($\|\mathcal{Z}\| = 30$)	1.820± 0.029	0.148 ± 0.022	0.219±0.031

the 25000 testing steps. The results from the 21 trials are used to calculate the mean and standard deviation for the RPR in each setting of $|\mathcal{Z}|$, which are reported in Table 2, along with the results of U-trees cited from [Zheng and Cho, 2011].

The online RPR performs comparably as U-trees on Shuttle and better on 4×4 Grid. The RPR performs worse than U-trees on Cheese Maze for the given budget of 75000 steps of learning. However, we notice that the RPR's performance can catch up after additional learning steps. Since the decision states in the RPR correspond to the nodes in U-trees, it is interesting to compare $|\mathcal{Z}|$ used by the RPR and the nodes generated by U-trees, which are shown as the parenthesized numbers in Table 2. As can be seen, the RPR generally needs less decision states to achieve superior performance, while U-trees generate more nodes than necessary.

To compare to iPOMDP [Doshi-Velez *et al.*, 2009] and policy prior [Doshi-Velez, 2010], we follow their experimental setting and report the cumulative rewards as a function of the learning step (3500 steps for Hallway, 5000 steps for Tag). The results are summarized in Table 3, which show that online RPR achieves significantly larger accumulative rewards than iPOMDP, and performs comparably to the policy prior methods which utilize expert information. These results demonstrate the potential benefits of learning a policy directly from the episodes, as opposed to solving a (possibly inaccurate) POMDP estimated from the episodes.

The MDP-EM method learns a reactive policy (it does not memorize past observations). To make the comparison fair, we consider the RPR policy conditioning on different lengths of history. The policy evaluation procedure is the same as described in Section 4.1, with the MDP-EM policy provided by the authors of [Vlassis and Toussaint, 2009]. The results are plotted in Figure 2, where $|h|$ indicates the length of history, e.g. $|h| = 1$ means only the current observation is used, which makes the RPR a reactive policy as the MDP-EM. As shown by Figure 2, the online RPR performs generally better than the MDP-EM, and the performance increases as $|h|$ increases from one to three, and then becomes saturated

Table 3: A comparison of online EM-RPR to iPOMDP and Policy Prior (PP), in terms of cumulative reward.

Problem	RPR (10)	RPR (40)	iPOMDP	PP-2	PP-3
Hallway	6.0±2.8	3.0±1.3	0.2	1.6	6.6
Tag	-4908±95	-4987±13	-16000	-7400	-3500

when longer histories are used. The improvement achieved by the RPR with $|h| > 1$ is expected, considering the MDP-EM uses only the current observation while the RPR with $|h| > 1$ also uses past actions and observations. However, even with $|h| > 1$, the RPR can also achieve significant improvements; the improvements can be attributed to the reward re-computation in (4) which, as discussed in the introduction and shown in (7), produce a weighted log-likelihood function whose maximization gives an improved policy.

5 Conclusions

We have presented an online nested EM algorithm for learning the RPR, a parametric policy for RL in POMDPs. The online algorithm uses only the latest learning episode to update the policy's value in the outer-loop E-step and update the sufficient statistic in the inner-loop E-step, and computes the improved policy in closed-form given the sufficient statistic. The algorithm reduces both the time and memory complexity an order of magnitude, in comparison with the corresponding batch-mode algorithm. Under standard conditions on the learning rates, the online algorithm provably converges to the optimal batch-mode policy when the number of learning episodes becomes sufficiently large.

Experimental results on five benchmark POMDP problems show that: (i) the online RPR produces policies as good as the batch-mode RPR by using only a small fraction of the time; (ii) the stochastic nature of the online algorithm can help it to jump out of local optima; (iii) model-free methods have the potential to learn better policies than model-based methods; (vi) the RPR can produce high-quality policies by using fewer internal memory units than the U-tree; (v) the RPR has a performance increasing with the history length, and is a competitive reactive policy when the length is one. Future work includes online inference of the number of decision states and multi-agent online learning.

Acknowledgments This research was supported by the US Office of Naval Research (ONR), under the Active Transfer Learning (ATL) and Science of Autonomy programs.

References

[Aberdeen and Baxter, 2002] D. Aberdeen and J. Baxter. Scalable internal-state policy-gradient methods for POMDPs. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 3–10. Morgan Kaufmann Publishers Inc., 2002.

[Blackwell, 1965] D. Blackwell. Discounted dynamic programming. *Ann. Math. Stat.*, 36:226–235, 1965.

[Cai *et al.*, 2009] C. Cai, X. Liao, and L. Carin. Learning to explore and exploit in POMDPs. In *Advances in Neural Information Processing Systems 22*, pages 198–206, 2009.

[Cappé and Moulines, 2009] O. Cappé and E. Moulines. Online EM algorithm for latent data models. *Journal of the Royal Statistical Society Series B*, 71(3):593–613, 2009.

[Delyon *et al.*, 1999] B. Delyon, M. Lavielle, and Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

[Dempster *et al.*, 1977] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

[Doshi-Velez *et al.*, 2009] F. Doshi-Velez, D. Wingate, N. Roy, and J. Tenenbaum. The infinite partially observable markov decision process. In *Advances in Neural Information Processing Systems 22*, pages 477–485, 2009.

[Doshi-Velez, 2010] F. Doshi-Velez. Nonparametric bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems 23*, pages 532–540. 2010.

[Grady *et al.*, 2013] D. Grady, M. Moll, and L. Kavraki. Automated model approximation for robotic navigation with pomdps. *Preprint*, Department of Computer Science, Rice University, 2013.

[Jordan and Jacobs, 1994] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[Kaelbling *et al.*, 1998] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[Li *et al.*, 2009] H. Li, X. Liao, and L. Carin. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 10:1131–1186, 2009.

[Liu *et al.*, 2011] M. Liu, X. Liao, and L. Carin. The infinite regionalized policy representation. In *The 28th International Conference on Machine Learning (ICML)*, pages 769–776, 2011.

[McCallum, 1995] R. A. McCallum. *Reinforcement Learning with Selective Attention and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

[Meuleau *et al.*, 1999] N. Meuleau, L. Peshkin, K. Kim, and L. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 427–43, San Francisco, CA, 1999. Morgan Kaufmann.

[Poupart and Vlassis, 2008] P. Poupart and N. Vlassis. Model-based Bayesian reinforcement learning in partially observable domains. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[Sondik, 1978] E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

[Sutton and Barto, 1998] R. Sutton and A. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

[Vlassis and Toussaint, 2009] N. Vlassis and M. Toussaint. Model-free reinforcement learning as mixture learning. In *The 26th International Conference on Machine Learning (ICML)*, pages 1081–1088, 2009.

[Wierstra and Wiering, 2004] D. Wierstra and M. Wiering. Utile distinction hidden Markov models. In *Proceedings of the International Conference on Machine Learning*, 2004.

[Zheng and Cho, 2011] L. Zheng and S. Cho. A modified memory-based reinforcement learning method for solving pomdp problems. *Neural Process Lett*, (33):187–200, 2011.