Optimistic Partially Observable Reinforcement Learning

1 Introduction

2 Background

2.1 Overview

We consider a finite POMDP defined as $(S, A, Z, \mathcal{R}, T, O, R, b_0, \gamma, H)$, where

- S is a finite set of states,
- A is a finite set of actions,
- Z is a finite set of observations,
- $\mathcal{R} \subset \mathbb{R}$ is a finite set of rewards,
- $T : S \times A \times S \to [0, 1]$ represents the transition model $T(s, a, s') = p(s'|s, a)$
- $O : S \times A \times Z \to [0, 1]$ represents the observation model $O(s', a, z) = p(z|s', a)$,
- $R : S \times A \times \mathcal{R} \to [0, 1]$ represents the reward model $R(s, a, r) = p(r|s, a)$,
- $b_0 : S \to [0, 1]$ is the initial state distribution,
- $H \in \mathbb{N}$ is the episode length, and
- $\gamma \in (0, 1)$ is a discount factor.

We will use $T(s, a, \cdot)$, $O(s, a, \cdot)$, $R(s, a, \cdot)$, and $b_0(\cdot)$ to refer to the probability distributions associated with each parameter.
Reinforcement learning consists of an agent interacting with a POMDP over time. Time is discretized into time-steps, and at each time-step, the agent first chooses and performs an action from $a \in A$, then the POMDP makes a state transition according to the transition model from the current state s to the next state $s' \sim T(s, a, \cdot)$ and outputs an observation $z \sim T(s', a, \cdot)$ and a reward $r \sim R(s, a, \cdot)$. The initial state is chosen according to the initial state distribution: $s \sim b(\cdot)$.

The key difference between partially observable RL and the more well-studied standard RL where the agent acts in a (fully observable) Markov decision process (MDP) is that the agent does not get to observe the hidden state s, except through the stochastic observation z. In most cases, the reward r also gives us information about the hidden state—to be precise, the reward is determined by the hidden state we *transitioned from* (s) and the action just taken (a), whereas the observation is determined by the reward we *transitioned to* (s') and the action just taken (a). Since the agent does not know the hidden state, the agent will maintain a **belief** of the hidden state $b : S \to [0, 1]$, where $b(s)$ is a probability that represents the belief that I am in state s, which is updated in a Bayesian fashion. We assume the agent's initial belief is the same as the initial state distribution b_0.
A **policy** π is a function that maps beliefs over states to actions. In the RL setting, the parameters of transition, observation, and reward models—or some subset of them—are unknown. The goal of RL is to use the data seen so far (i.e. the sequence of actions taken, observations seen, and the values of the rewards obtained) to learn a policy that maximizes long-term rewards. We

assume the agent continues for infinite time, but the rewards are discounted by the γ, so the total reward obtained is finite; that is we want to maximize: $\sum_{t=1}^{\infty} \gamma^t r_t$. The optimal policy π^* is the policy that maximizes long-term rewards. The algorithms we consider are model-based, meaning they will try to infer the model parameters and use those in some fashion to find and execute a policy.

We will be focusing on episodic domains in particular. In an episodic domain, after H steps the state resets to a state distributed according to b_0 (and the belief is likewise reset to b_0). In the POMDP literature, a domain can reset when some state is reached or some reward is obtained. For example, if we were studying a robot traversing a maze, we could say the domain resets when the robot reaches the goal. The episodic nature of a such a domain is natural. We can't expect the robot to do well in the maze on it's first try; we have to give it several chances to learn a good policy for getting to the goal state. The objective of RL in this case might be to have the robot learn how to get to the goal optimally in as few training rounds as possible. However, we are only considering fixed-horizon episodic domains. We make this assumption so we can obtain the sample complexity result discussed below. It is an open question if we can obtain a similar result while relaxing this assumption. However, the version of the algorithm we use in practice can just as easily be applied to non-episodic or non-fixed length episodic domains.

2.2 Optimistic RL

The key idea behind our algorithm is to use optimism in the face of uncertainty to guide exploration, meaning that so long as the agent is uncertain about the parameters, the agent will assume the parameters of the POMDP are those which yield the best possible rewards and choose an action accordingly. If that action did poorly, the agent will reduce its uncertainty of the relevant parameters, making it realize other parameters might be more reasonable; if that action did well, the agent can try to exploit its current policy without bothering with other ones. This enables the agent to make choices it would otherwise avoid if it were to greedily exploit the policy it currently thinks is best; of course, it does so at the expense of occasionally making some suboptimal choices. Optimism under uncertainty has been shown to be successful approach in the fully observable RL setting, both practically and theoretically [**?**]. In particular, one state of the art algorithm in fully observable RL is model-based interval estimation (MBIE) algorithm for MDPs, which estimates confidence intervals over model parameters and plans optimistically with respect to those confidence intervals. This algorithm has a PAC-MDP sample complexity guarantee as well as experimentally outperforming other standard MDP RL algorithms (that also have PAC-MDP guarantees) [**?**]. The algorithm we develop in this paper is heavily influenced by the MBIE algorithm, but with modifications to make it work in the partially observable setting.

3 Optimistic Partially Observable RL Algorithm

We split up the RL problem into two parts: the first part is to learn the confidence intervals for the transition and observation parameters (**learning**), and the second part is to use those confidence intervals for optimistic online planning (**planning**).

In this section, we first give a general description of an optimistic partially observable RL algorithm. This algorithm will be general enough such that any learning algorithm and planning algorithm that meet the requirements we discuss can be used. This general algorithm is also given in Algorithm 1. For the rest of the paper, we will use a particular instantiation of this algorithm to obtain theoretical results, and we will use another (albeit similar) instantiation of the algorithm to obtain experimental results. The planning algorithm is the same for both instantiations, so we will give the details of the planning algorithm at the end of this section after describing the general algorithm.

3.1 Learning

For learning, we can use any algorithm that estimates confidence intervals over the model parameters. Denote the lower bound for the transition parameter by $\underline{T}(s, a, s')$ and the upper bound by $\overline{T}(s, a, s')$. Similarly for the observation parameters we have $\underline{O}(s', a, z)$ and $\overline{O}(s', a, z)$ and for the reward parameters we have $\underline{R}(s, a, r)$ and $\overline{R}(s, a, r)$. Note that these bounds can be probabilistic. In

other words, a $1-\alpha$ confidence interval for the transition parameters is one in which the probability that

$$\underline{T}(s,a,s') \leq T(s,a,s') \leq \overline{T}(s,a,s')$$

is at least $1-\alpha$. Similarly, a $1-\alpha$ confidence interval for the observation and reward parameters is one in which the probability that

$$\underline{O}(s',a,z) \leq O(s',a,z) \leq \overline{O}(s',a,z)$$

and

$$\underline{R}(s,a,r) \leq R(s,a,r) \leq \overline{R}(s,a,r)$$

is at least $1-\alpha$ respectively. Note that we can now construct a joint confidence interval over models, which is the set of all valid probability distributions for the transition, observation, and reward models between the lower and upper bounds. For example, the vector $\underline{T}(s,a,\cdot)$ is not necessarily a probability distribution as $\sum_{s'} \underline{T}(s,a,s') \leq 1$ and likewise for the vector $\overline{T}(s,a,\cdot)$, where $\sum_{s'} \overline{T}(s,a,s') \geq 1$. Of course, a concise way to describe this joint confidence interval over models is using the upper and lower bounds for each model parameter.

While in theory we require valid confidence intervals in order to obtain sample complexity results, in practice, any intervals will do, so long as the lower bound is less than the upper bound. In practice the size of the confidence interval would be a tuning parameter; if the confidence intervals are valid, α could be the tuning parameter.

3.2 Planning

The optimistic planning algorithm takes as input the confidence intervals outputted by the learning algorithm, and outputs the action. To this end the algorithm must find the optimistic model parameters in the confidence intervals; that is, the model in the confidence intervals whose optimal policy yields the highest rewards. In some sense, we can think of it as if our algorithm thinks it can set the underlying model parameters of the POMDP to be anything within the model confidence intervals; the algorithm will then try to find the parameters that yield the highest reward. Note that we cannot simply apply some procedure that finds this model, and the apply standard POMDP planning to find the optimal policy; since we want to find the model that maximizes rewards, we must do planning in order to find that model. Once an optimistic model and associated policy are found, we can do a belief update using the optimistic model, and then find the next action to take by applying the policy to our current belief. Any algorithm that satisfies this description of optimistic planning can be used. For example, Ni and Liu 2012 give an optimistic planning algorithm that uses finite state controllers [**?**]. There approach is unscalable in practice, so we use extend use an optimistic planning algorithm based on point-based value iteration (PBVI) [**?**]. However, aside from how we represent the policies, our algorithm has a lot in common with the one in Ni and Liu 2012. We present this algorithm in detail in the remainder of this section.

3.3 Point-Based Optimistic Planning Algorithm

The planning algorithm we use is based on point-based value iteration (PBVI), an approximate POMDP planning algorithm [**?**]. In this algorithm we maintain a set of α-vectors, where $\alpha(s)$ be the expected reward obtained by executing a particular policy starting from state s (i.e. assuming we know for certain we start in state s, but thereafter we don't see the hidden state, except through observations). Every α-vector has a corresponding policy tree, which specifies what action to take after receiving a particular observation at each time-step, and we use these α-vectors to compare different policy trees. Our expected reward from executing a particular policy with associated α-vector α_π is given by

$$\sum_s b(s)\alpha_\pi(s)$$

α-vectors are used to approximate the optimal value function, which is the function that gives the the expected reward from executing the optimal policy for a given belief. If we had an α-vector that maximized the dot-product above for every possible belief, then our planning algorithm would simply need to choose the α-vector associated with our current belief and execute its associated

policy. However, this is intractable. To get around this issue, PBVI maintains only a small number of possible beliefs (a belief is a probability distribution over the states), which reduces the number of α-vectors that it needs to consider; of course, this in turn makes the algorithm an approximate algorithm, and the more beliefs we maintain, the better the results, but for small domains, a small number of beliefs may be sufficient to find the best optimistic policy given the models [**?**]. Thus we maintain a set of belief points B and the corresponding set of α-vectors M. At every time-step, we will perform an optimistic belief point backup on some (possibly all) of the belief points, to get new updated α-vectors for those belief points (see Algorithm 2). PBVI expands the set of belief points over time [**?**], but since we only consider simple POMDPs in our experiments, we do not need to keep adding new belief points; we always use a grid of evenly-spaced belief points in the belief space. However, our algorithm can easily be extended to expanding the set of belief points in a fashion similar to PBVI.

Of course all of this is only useful if we have a way of computing the α-vectors, especially a way of doing so optimistically with respect to the given confidence intervals. We do this using the optimistic belief point backup described below.

3.4 Optimistic Belief Point Backup

A backup refers to creating a new set of α-vectors. We can start with a set of α-vectors by assuming we will only take one action, meaning we want to choose the action that maximizes the immediate reward given our belief. If we do this for a set of belief points B, we now have a set of *one-step* α-vectors. Of course, we don't just want to take one action, rather we want to take a sequence of infinitely many actions. Thus we now apply a backup to this set of one-step α-vectors, to get a set of two-step α-vectors and so on. Of course, we cannot repeat this process ad infinitum, but since our rewards are discounted by γ, doing only $\frac{1}{1-\gamma}$ backups will give us a good approximation; in practice, we can even get by with less. We now describe the optimistic belief point backup (which is also given in Algorithm 2).

For each new α-vector α_a is associated with a policy tree consisting of a root action a followed by the policy tree of an existing α-vector for each possible observation. This means we have a total of $|A||M|^{|Z|}$ possible new α-vectors. To compute α_a consisting of root action a, followed by the policy tree of α_z, the α-vector that we pick for observation z, we need to evaluate the following (see Algorithm 3):

$$\begin{aligned}\alpha_a(s) &= \max_R \mathbb{E}R(s,a,\cdot) + \gamma \max_T \max_O \sum_{s'} \left[T(s,a,s') \sum_z O(s',a,z)\alpha_z(s') \right] \\ &= \max_R \sum_r r \cdot R(s,a,r) + \gamma \max_T \sum_{s'} \left[T(s,a,s') \max_O \sum_z O(s',a,z)\alpha_z(s') \right]\end{aligned}$$

The optimistic property comes from the fact that we take the maximum over all possible instantiations of the model within the confidence intervals.

Note that we can compute the maximization with respect to the reward parameters independently of the other two maximizations, since the others don't depend on the rewards. Similarly, we can compute the inner maximization with respect to the observation parameters independently of the outer maximization with respect to the transition parameters, since the inner maximization does not depend on the transition parameters. Once we have done that, we can then do the outer maximization with respect to the transition parameters. To do the maximization with respect to the rewards, note that we want to allocate the highest probability to the largest reward, the second highest probability to the second largest reward and so on. Recall that $R(s,a,r)$ needs to be at least $\underline{R}(s,a,r)$ for each r, so we first initialize all of the probabilities accordingly. We then do the following for each reward r_i in descending order: change the probability of $R(s,a,r_i)$ from $\underline{R}(s,a,r_i)$ to $\overline{R}(s,a,r_i)$ if it does not violate $\sum_r R(s,a,r) \leq 1$; if it does violate that condition, then we simply set $R(s,a,r_i) \leftarrow$

$1 - \sum_{r \neq r_i} R(s, a, r)$. Note that this clearly results in a probability distribution that maximizes the expected reward. We can follow the same procedure for the other two maximizations.

Algorithm 1 Optimistic Partially Observable RL

$h \leftarrow [\,]$ # The history starts out with nothing.
while True **do**
  $(\underline{T}, \overline{T}, \underline{O}, \overline{O}, \underline{R}, \overline{R}) \leftarrow$ LearnCIs(h)
  $a \leftarrow$ OptimisticPlanning($\underline{T}, \overline{T}, \underline{O}, \overline{O}, \underline{R}, \overline{R}, h$) # Need h to compute belief
  $h \leftarrow$ TakeAction(a) # Adds new action, observation, and reward to h
end while

Algorithm 2 Optimistic Belief Point Backup

Input: belief b, set of existing α-vectors M
$v \leftarrow -\infty$
for $a \in A$ **do**
  for choice of α_z for each observation z **do** # pick an element of $M^{|Z|}$
    $\alpha_a \leftarrow$ optimistic one-step policy evaluation of a and α_z
    if $\sum_s b(s)\alpha_a(s) > v$ **then**
      $v \leftarrow \sum_s b(s)\alpha_a(s)$
      $a^* \leftarrow a$
    end if
  end for
end for
Output: a^*

3.5 Optimistic Belief Update

Once we have an optimistic instantiation of the model parameters, we can do a regular belief update on the optimistic models. We recompute the belief from scratch at each time-step after obtaining the optimistic model, assuming that model was the true model from the beginning.

4 Analysis

In this section we will first discuss the particular learning algorithm we will use, which is based on spectral learning. We will then define the PAC setting, and finally we will prove our algorithm is a PAC-POMDP algorithm.

4.1 Method of Moments Learning Algorithm

The optimistic planning algorithm takes as input the confidence intervals outputted by the learning algorithm, and outputs the action. This algorithm can apply any POMDP planning procedure To this end we can rely on algorithms for learning the parameters of hidden Markov models (HMMs). A HMM is a model with transitions and observations but no actions or rewards; the parameters of a POMDP for a fixed action can be expressed as a HMM with a new observation model that combines observations and rewards (since observations depend on the state that was transitioned to and rewards depend on the state we transitioned from, we can combine the current reward and the previous observation into a new joint observation). Thus we can use any HMM inference algorithm to learn the parameters of a POMDP. Unfortunately learning the parameters of a HMM is formally hard problem under generic conditions [**?**]. Thus, a common algorithm to estimate HMM parameters is the expectation maximization (EM) algorithm, but this algorithm is not guaranteed to asymptotically reach the global optimum. Moreover it does not yield confidence intervals over the estimated parameters. Hence, we cannot use it to get any theoretical guarantees. A new approach to learning HMM parameters is using spectral learning method of moments approaches. These algorithms often yield admissible confidence intervals, which makes them a useful choice for our RL algorithm. For purposes of the theory below we will use one such algorithm that yields confidence intervals of the

Algorithm 3 Optimistic One-Step Policy Evaluation

Input: action a, choice of α_z for each z
for $s \in S$ **do** # rewards max
  $R(s,a,r) \leftarrow \underline{R}(s,a,r), \forall r \in \mathcal{R}$
  $D \leftarrow$ descending_sort($\mathcal{R}$)
  while $\sum_z R(s,a,r) < 1$ **do**
    $r' \leftarrow D$.pop()
    $R(s,a,r') \leftarrow R(s,a,r') + \min(1 - \sum_r R(s,a,r), \overline{R}(s,a,r') - \underline{R}(s,a,r'))$
  end while
end for
for $s' \in S$ **do** # inner max
  $O(s',a,z) \leftarrow \underline{O}(s',a,z), \forall z \in Z$
  $D \leftarrow$ the order of observations such that $\alpha_z(s')$ is in descending order
  while $\sum_z O(s',a,z) < 1$ **do**
    $z' \leftarrow D$.pop()
    $O(s',a,z') \leftarrow O(s',a,z') + \min(1 - \sum_z O(s',a,z), \overline{O}(s',a,z') - \underline{O}(s',a,z'))$
  end while
  $\tilde{\alpha}(s') \leftarrow \sum_{z'} p(z'|s',a)\alpha_{z'}(s')$
end for
for $s \in S$ **do** # outer max
  $T(s,a,s') \leftarrow \underline{T}(s,a,s'), \forall s' \in S$
  $D \leftarrow$ the order of states such that $\tilde{\alpha}(s')$ is in descending order
  $s' \leftarrow$ first element of D
  while $\sum_{s'} T(s,a,s') < 1$ **do**
    $s'' \leftarrow D$.pop()
    $T(s,a,s'') \leftarrow T(s,a,s'') + \min(1 - \sum_{s'} T(s,a,s'), \overline{T}(s,a,s'') - \underline{T}(s,a,s''))$
  end while
  $\alpha_a(s) \leftarrow \sum_r r \cdot R(s,a,r) + \gamma \sum_{s'} T(s,a,s')\tilde{\alpha}(s')$
end for
Output: α_a

form $C/\sqrt{n}$ where n is the number of steps taken using the HMM, and the constant C depends on a number of . For POMDPs, it would be natural to expect this result to extend to the confidence intervals taking the form $C/\sqrt{n_a}$, where n_a is the number of times the RL algorithm took action a and C is a constant in terms of $\sqrt{n_a}$ (i.e. C does not change during the course of RL but it depends on a number of parameters of the POMDP); there is actually a slight caveat in that not all instances of taking action a can necessarily be used, but for now we will assume the confidence intervals take this form. In fact, the aforementioned caveat is just one of several difficulties with using the method of moments as our inference technique; we discuss these difficulties below.

4.2 Method of Moments Caveats

As mentioned, there are a few caveats with using the method of moments as our inference technique (which is why we use EM for our experiments). First, since learning the parameters of a HMM is formally hard, there must be some constraints on the HMM, which allows the method of moments technique to find statistically consistent estimates. The constraints we need are of this form:

Condition 1. *The matrices $T(\cdot, a, \cdot)$, $O(\cdot, a, \cdot)$, and $R(\cdot, a, \cdot)$ are all full rank.*

The next set of caveats fall from adapting the method of moments technique to work on POMDPs. Recall that the method of moments uses triples of observations as atomic units of data. Suppose part of our history is as follows $(a_0, z_1, a_1, z_2, a_2, z_3, a_3)$—this could potentially be used as a "data point" for method of moments. To learn $T(\cdot, a, \cdot)$, we can only use the trajectory above if a_1 is the same action for all the data we are using, a_2 is the same action for all the data we are using, and a_3 is the same action for all the data we are using (but a_1, a_2, and a_3 can be different from each other). Aside from limiting the amount of data we can use, the true problem arises when we take into account that the agent chooses which actions to take. Since we can only use a certain sequences of actions for learning, we could be biasing the learned parameters—for example, suppose the agent only takes actions a_1, a_2, and a_3 in succession, when it (thinks it's) is in part of the state space, but in other parts of the state space still takes action a_2. When we learn the transition parameters for a_2, we would not be learning a good model for it in the other parts of the state space. This bias voids the theoretically valid confidence intervals we need to obtain the sample complexity of our algorithm. To thwart this issue, we can only use samples from the beginning of each episode, as long as b_0 assigns some probability of starting in each state. Note that this requires us to use episodic domains, but we would need this assumption (or a similar one) regardless for our sample complexity result. In particular, we need a fixed episode. In fact, we can't use problems that have episodes embedded into the description of the POMDP for several reasons.

For example, a classic POMDP domain is the Tiger domain, where an agent is faced with two doors, one of which has a tiger on the other side. The agent can listen to the two doors to get a noisy observation of where the tiger is, and can listen as many times as she wants in order to refine her belief of the tiger's position before opening a door. The rewards naturally incentivize the agent to try to open the door without the tiger with high probability. After opening a door, the problem resets. This reset is incorporated into the POMDP by having the agent uniformly transition to another state, and obtain a uniform random observation, giving no information about the state transitioned to. Note that if the transition or observation matrix is uniform random, the method of moments cannot learn it, because Condition 1 is violated. However, we may still be able to learn the other parameters (and naturally assume the parameters that reset the state are known). An additional problem lies in that we can only use triples of observations without opening a door between them. Under the optimal policy, the number of times the agent listens before opening the door depends on the observations—if an agent gets conflicting observations, the agent will need to listen more times—sometimes the agent will listen a number of times divisible by three, and other times the agent will not. Since we disregard all the times the agent listens a number of times not divisible by three, which would be due to how many conflicting observations there were, we are significantly biasing our estimate of the observation parameters. In fact, using the traditional Tiger parameters, the optimal policy would often only listen twice before opening the door; we can never use this data to learn the parameters, which is already a significant bias.

4.3 Probably Approximate Correct Setting

There are several metrics one can use to make guarantees about the performance of an RL algorithm. Strehl and Littman 2008 gives a survey of some of these metrics in the MDP setting, including a discussion of probably approximate correctness (PAC) as a criterion for evaluating the performance of an algorithm. A PAC algorithm is one that achieves near-optimal reward on most time steps with high probability. Notice that such an algorithm can get arbitrarily bad reward on some time-steps, but it will receive near optimal reward on the rest of the time steps; we have to allow for this, since the agent starts out knowing nothing about its environment. There are a number of ways to formalize the PAC notion; we will adapt the formalization given by Kakade. First define the sample complexity (of exploration) of a partially observable RL algorithm $\mathcal{A}$ as the number of time-steps for which the agent does not get near-optimal reward, that is the number of time steps t where

$$V^{\pi^*(b_t^*)} - V^{\mathcal{A}_t}(b_t^*) \geq \epsilon$$

where $\mathcal{A}_t$ is the policy of the algorithm at time-step t, π^* is the optimal policy, andb_t^* is the belief as computed by the true models.

4.4 Sample Complexity Result

Need to fill in proof.

5 Experiments

These problems make it such that it would be practically very inconvenient to use the method of moments. Thus in practice we will actually use EM to learn the POMDP parameters. Note that our main motivation for using EM is that the method of moments technique places many inconvenient constraints on the algorithm and on the domains in which it applies; however, it is also worth noting that the method of moments practically performs much worse than EM, so EM appears preferable in practice regardless.

However, as we already mentioned EM does not give us confidence intervals, so we need a way to get around this issue. There are several ways to get approximate confidence intervals for EM such as using the bootstrap technique [**?**]. We take an alternative approach of using pseudo-confidence intervals—that is we do not guarantee that the true parameters fall in these confidence intervals with probability $1 - \alpha$ even asymptotically or approximately. In fact, since EM finds only local optima and these intervals will be around the solutions of EM, they can be quite off, but we expect them to perform well in practice, when they find the global optimum, or when the local optima are close to the global optimum. Despite being technically incorrect, we will still refer to these intervals as confidence intervals for simplicity. To explain these confidence intervals, note that in POMDPs, we do not know how many times the agent has visited a certain state, but if we did (i.e. if we are actually acting in an fully observable Markov Decision Process), we could use confidence intervals based on Hoeffding's inequality. That is, if we let

$$\epsilon_{T_{(s,a,\cdot)}} = \sqrt{\frac{1}{n_{s,a}} \log \frac{\alpha}{2}}$$

then our confidence interval for T is defined by setting:

$$\overline{T}(s,a,s') = \hat{T}(s,a,s') + \epsilon_{T_{(s,a,\cdot)}} \text{ and } \underline{T}(s,a,s') = \hat{T}(s,a,s') - \epsilon_{T_{(s,a,\cdot)}}$$

where $\hat{T}(s,a,s')$ is our estimate of the transition parameters (obtained by EM). We can similarly define confidence intervals for the observation and reward parameters using analogously defined $\epsilon_{O_{(s,a,\cdot)}}$ and $\epsilon_{R_{(s,a,\cdot)}}$. To account for the fact that we don't know how many times we visited a certain state-action pair, we will replace $n_{s,a}$ with $\gamma_{s,a}$ which is EM's estimate of how many times we visit a certain state-action pair. Since there are no guarantees of how $\gamma_{s,a}$ approximates $n_{s,a}$, we cannot make any formal guarantees about our confidence intervals as mentioned; however, they seem intuitively appealing. They are also much quicker to compute than using bootstrap—since there is hardly any additional computation once we have $\gamma_{s,a}$, which EM gives us—and they perform well in our experiments.

6 Appendix

6.1 Method of Moments Confidence Intervals

The method of moments algorithm that we use (as far as the theory is concerned) for learning the models gives confidence intervals that we cannot use off-the-shelf. In particular, there are four manipulations we need to make to have the confidence intervals be useful for the theoretical analysis that follows.

1. The method of moments algorithm gives confidence intervals in terms of the L2 norm of each distribution, whereas we want confidence intervals that bound each parameter individually
2. The method of moments does not necessarily yield valid probability distributions (i.e. some of the parameters might be negative and the sum of the "probabilities" need not be one); whereas we, of course, need valid probability distributions. To get around this we would like to replace all negative probabilities with zero and normalize the resulting distribution. We need to show that confidence intervals still exist for the resulting parameters.
3. We treat the observations and rewards together as a joint observation in the method of moments algorithm, which results in learning joint parameters. We need separate confidence intervals for the observation and reward parameters.
4. The method of moments does not give confidence intervals for the transition parameters directly; it gives confidence intervals for $OT(s,a,\cdot) = \sum_{s'} O(s',a,\cdot)T(s,a,s')$. We would like to use these and the fact that we have confidence intervals for the observation parameters, to derive confidence intervals for the transition parameters.

We resolve each of these four in turn.

6.1.1 Resolving L2 Norm Confidence Intervals

Assume for the observation parameter we have confidence intervals of the form

$$||O(s',a,\cdot) - \hat{O}(s',a,\cdot)|| \leq \epsilon \tag{1}$$

We want to use these to get confidence intervals for $|O(s',a,z) - \hat{O}(s',a,z)|$. This is straightforward, since:

$$|O(s',a,z) - \hat{O}(s',a,z)| \leq \sqrt{\sum_z |O(s',a,z) - \hat{O}(s',a,z)|^2} = ||O(s',a,\cdot) - \hat{O}(s',a,\cdot)|| \leq \epsilon \tag{2}$$

6.1.2 Valid Probability Distributions

To turn the probability distributions that method of moments algorithm gives out, we must first replace all negative probabilities with zero. Let $\hat{p}$ correspond to some probability distribution given by method of moments (e.g. $O(s,a,\cdot)$), and p corresponding to the true probability distribution. Notice that if any component of p is negative, replacing it by zero, will only decrease the value of

$$||p - \hat{p}|| = \sqrt{|\sum_x p(x) - \hat{p}(x)|^2}$$

since $p(x) > 0$ for all x, so $\hat{p}(x)$ can only be closer. So heretofore whenever referring to estimated parameters $\hat{p}(x)$ we assume they are at least zero.

Now consider normalizing p. Let X be the domain of p. Since we know that $|p(x) - \hat{p}(x)| \leq \epsilon$ for all x, we also have that

$$\|p - \hat{p}\|_1 = \sum_{x \in X} |p(x) - \hat{p}(x)| \leq |X|\epsilon \tag{3}$$

[Note that Emma says $|X|\epsilon$ could be replaced with the tighter $\sqrt{|X|}\epsilon$ in equation (3), but I haven't verified this yet, so I'm going to leave it as is for now, and change it when I see why it works.] Since p is a valid probability distribution, we have that

$$\|p\|_1 = \sum_x p(x) = 1 \tag{4}$$

Thus, using the inverse triangle inequality and equation (3),

$$\begin{aligned} & | \, \|\hat{p}\|_1 - \|p\|_1 \, | \leq \|p - \hat{p}\|_1 \leq |X|\epsilon && (5) \\ \Rightarrow \quad & \|p\|_1 - |X|\epsilon \leq \|\hat{p}\|_1 \leq \|p\|_1 + |X|\epsilon && (6) \\ \Rightarrow \quad & 1 - |X|\epsilon \leq \|\hat{p}\|_1 \leq 1 + |X|\epsilon && (7) \end{aligned}$$

where equation (7) follows from substituting in equation (4).
Thus, if $p(x) > \hat{p}(x)$,

$$\begin{aligned} \left| p(x) - \frac{\hat{p}(x)}{\|\hat{p}(x)\|_1} \right| \leq & \quad \left| p(x) - \frac{\hat{p}(x)}{1 + |X|\epsilon} \right| && (8) \\ = & \quad \left| \frac{(1 + |X|\epsilon)p(x) - \hat{p}(x)}{1 + |X|\epsilon} \right| && (9) \\ \leq & \quad \frac{|p(x) - \hat{p}(x)|}{1 + |X|\epsilon} + \frac{|X|\epsilon p(x)}{1 + |X|\epsilon} && (10) \\ \leq & \quad \frac{(1 + |X|)\epsilon}{1 + |X|\epsilon} && (11) \\ < & \quad (|X| + 1)\epsilon && (12) \end{aligned}$$

where inequality (8) follows from the fact that $p(x) > \hat{p}(x) \geq \frac{\hat{p}(x)}{\|\hat{p}(x)\|_1} \geq \frac{\hat{p}(x)}{1 + |X|\epsilon}$, inequality (10) follows from the triangle inequality, and inequality (11) follows from the fact that $|p(x) - \hat{p}(x)| \leq 1$ and the fact that $p(x) \leq 1$, and inequality 12 follows by removing $|X|\epsilon$ from the denominator, which can only increase the value of the fraction.
If $p(x) < \hat{p}(x)$

$$\begin{aligned} \left| p(x) - \frac{\hat{p}(x)}{\|\hat{p}(x)\|_1} \right| \leq & \quad \left| p(x) - \frac{\hat{p}(x)}{1 - |X|\epsilon} \right| && (13) \\ = & \quad \left| \frac{(1 - |X|\epsilon)p(x) - \hat{p}(x)}{1 - |X|\epsilon} \right| && (14) \\ = & \quad \left| \frac{p(x) - \hat{p}(x) - |X|\epsilon p(x)}{1 - |X|\epsilon} \right| && (15) \\ = & \quad \left| \frac{-\epsilon - |X|\epsilon p(x)}{1 - |X|\epsilon} \right| = \quad \frac{(|X| + 1)\epsilon}{1 - |X|\epsilon} \leq 2|X|\epsilon && (16) \end{aligned}$$

where the last inequality holds as long as $\epsilon \leq \frac{|X|-1}{|X|}$. Note that this won't always be the case. Thus, in our algorithm, we take into account that we don't have formal guarantees until this condition is met. Heretofore whenever referring to estimated parameters $\hat{p}(x)$ we assume they are normalized, and hence a valid probability distribution.

6.1.3 Disentangling Observations and Rewards

When we want to learn both observation parameters and the reward parameters, the method of moments does not directly give us confidence intervals over either. Rather, it gives us confidence intervals of the following form:

$$|O(s, a_1, z)R(s, a_2, r) - \hat{O}(s, a_1, z)\hat{R}(s, a_2, r)| \leq \epsilon$$

Notice that here the state s is the same but the actions are (possibly) different. This is because we are combining the observation from time t with the reward from time $t + 1$ so they are emitted from

the same state. It doesn't matter in terms of the method of moments algorithm that the actions are different, but we can only use data where action a_1 is taken followed by action a_2. For simplicity, below we will assume $a := a_1 = a_2$. That is to say we only use data where two consecutive actions are the same to learn these parameters.

From these confidence intervals, we want to obtain a similar confidence interval for each of the observation and reward parameters. For the observation parameters, we have,

$$|O(s,a,z) - \hat{O}(s,a,z)| = \left| \sum_r O(s,a,z)R(s,a,r) - \sum_r \hat{O}(s,a,z)\hat{R}(s,a,r) \right|$$
$$\leq \sum_r |O(s,a,z)R(s,a,r) - \hat{O}(s,a,z)\hat{R}(s,a,r)| \leq |R|\epsilon$$

Similarly, for the reward parameters,

$$|R(s,a,z) - \hat{R}(s,a,z)| = \left| \sum_z O(s,a,z)R(s,a,r) - \sum_z \hat{O}(s,a,z)\hat{R}(s,a,r) \right|$$
$$\leq \sum_z |O(s,a,z)R(s,a,r) - \hat{O}(s,a,z)\hat{R}(s,a,r)| \leq |Z|\epsilon$$

6.1.4 Deriving Transition Confidence Intervals

Again, the method of moments algorithm does not give us confidence intervals for the transition parameters directly. Instead it gives confidence intervals of the form:

$$\left| \sum_{s'} O(s',a,z)T(s,a,s') - \sum_{s'} \hat{O}(s',a,z)\hat{T}(s,a,s') \right| \leq \epsilon$$

(In reality, if we are learning the rewards, the $O(s',a,z)$ should be replaced by $O(s',a_1,z)R(s',a_2,r)$ as in section 4.1.3, but, without loss of generality, we will abuse notation and just write O. One can think of it as representing the joint observation parameters instead.)

First we will "get rid of" the approximation error for the observation parameter, since we know it is bounded by ϵ. Let $\Delta O(s',a,z) = \hat{O}(s',a,z) - O(s',a,z)$. Using the inverse triangle inequality, we have that,

$$\left| |\sum_{s'} O(s',a,z)(T(s,a,s') - \hat{T}(s,a,s'))| - |\sum_{s'} \Delta O(s',a,z)\hat{T}(s,a,s')| \right|$$
$$\leq \left| \sum_{s'} O(s',a,z)(T(s,a,s') - \hat{T}(s,a,s')) - \Delta O(s',a,z)\hat{T}(s,a,s') \right|$$
$$= \left| \sum_{s'} O(s',a,z)T(s,a,s') - \sum_{s'} \hat{O}(s',a,z)\hat{T}(s,a,s') \right| \leq \epsilon$$

Notice that since $|O(s',a,z) - \hat{O}(s',a,z)| \leq \epsilon$, we have that,

$$|\sum_{s'} \Delta O(s',a,z)\hat{T}(s,a,s')| \leq |S|\epsilon$$

Thus, we can conclude that

$$\left| \sum_{s'} O(s',a,z)(T(s,a,s') - \hat{T}(s,a,s')) \right| \leq (|S|+1)\epsilon$$

Let

$$\Delta T(s,a,\cdot) := T(s,a,\cdot) - \hat{T}(s,a,\cdot)$$

We will prove by way of contradiction that it must be the case that $|\Delta T(s,a,s')| \leq C\epsilon$ for all $s, s' \in S$, $a \in A$, and for some constant C. So assume that $|\Delta T(s,a,s')| > C\epsilon$. Notice that the parameters of T can (possibly) be functions of ϵ. Our assumption implies that some component of $\Delta T(s,a,\cdot)$ must not be $O(\epsilon)$. Let $\Delta_C T(s,a,\cdot)$ be the parts of each component of $\Delta T(s,a,\cdot)$ that are not $O(\epsilon)$. Now let

$$S_+ = \{s' | \Delta_C T(s,a,s') \geq 0\}$$

and let $S_- = S \backslash S_+$. Since $|\sum_{s'} O(s',a,z)\Delta T(s,a,s')| \leq (|S|+1)\epsilon$ know that for all s'

$$\sum_{s'} O(s',a,z)\Delta_C T(s,a,s') = 0$$

Thus, it follows that,

$$\sum_{s' \in S_+} O(s',a,z)\Delta_C T(s,a,s') = \sum_{s' \in S_-} O(s',a,z)\Delta_C T(s,a,s')$$

Since this is true for all Z. Let s^* be any element in S_+. We can see that

$$O(s^*,a,\cdot) = \frac{1}{\Delta_C T(s,a,s^*)} \left(\sum_{s' \in S_-} O(s',a,\cdot)\Delta_C T(s,a,s') - \sum_{s' \in S_-} O(s',a,\cdot) \right)$$

This violates Condition 1, specifically that $O(\cdot,a,\cdot)$ must be full rank. Thus we have proven that $|\Delta T(s,a,s')| \leq C\epsilon$ for all $s, s' \in S$, $a \in A$, and some constant C.

Test Emma's theory:

$$\sum_{s'} O(s',a,z)(T(s,a,s') - \hat{T}(s,a,s'))$$