Reinforcement Learning with State Observation Costs in Action-Contingent Noiselessly Observable Markov Decision Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many real-world problems that require making optimal sequences of decisions under uncertainty involve costs when the agent wishes to obtain information about its environment. We design and analyze algorithms for reinforcement learning (RL) in Action-Contingent Noiselessly Observable MDPs (ACNO-MDPs), a special class of POMDPs in which the agent can choose to either (1) fully observe the state at a cost and then act; or (2) act without any immediate observation information, relying on past observations to infer the underlying state. ACNO-MDPs arise frequently in important real-world application domains like healthcare, in which clinicians must balance the value of information gleaned from medical tests (e.g., blood-based biomarkers) with the costs of gathering that information (e.g., the costs of labor and materials required to administer such tests). We develop a PAC RL algorithm for tabular ACNO-MDPs that provides substantially tighter bounds, compared to generic POMDP-RL algorithms, on the total number of episodes exhibiting worse than near-optimal performance. For continuous-state, continuous-action ACNO-MDPs, we propose a novel method of incorporating observation information that, when coupled with modern RL algorithms, yields significantly faster learning compared to other POMDP-RL algorithms in several simulated environments.

1 Introduction

Reinforcement learning, which involves learning to make sequences of good decisions in stochastic environments, has had many impressive successes over the past decade [50, 56]. While many of the more visible successes have occurred in simulated environments, such as Atari video games [34, 48], reinforcement learning has increasingly been used to solve real-world problems [6, 36, 27, 31, 61, 37, 28, 41, 32, 25]. One key difference between reinforcement learning in simulated vs. real-world environments is that, in most simulated environments, the agent can fully observe the underlying state at each time step with no additional observation costs. Thus Reinforcement Learning (RL) algorithms that treat the underlying environment as a Markov decision process (MDP) can demonstrate considerable success in these environments, as there is no uncertainty in the state. In contrast, real-world environments frequently entail state observations that are noisy and/or costly, but optional. Environments in which state information at each time step can be noisy or missing are more appropriately modeled as partially observable Markov decision processes (POMDPs) rather than MDPs [20].

In this paper, we provide theory and algorithms for a special class of POMDPs in which state information is complete/noiseless when observed, but may be missing at any given time step if the agent chooses not to observe the state. We call this class of POMDPs *Action-Contingent Noiselessly*

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Observable MPDs (ACNO-MDPs), and to the best of our knowledge this paper is the first to consider such a setting. Given the disproportionate focus of RL research on simulated environments with trivial observation costs, it is perhaps not surprising that POMDPs with observation costs have been understudied. We claim, however, that ACNO-MDPs are an appropriate model for a number important real-world problem settings. We give two such examples, though many more exist.

ACNO-MDPs in healthcare. Clinicians in Intensive Care Units (ICUs) frequently have to make sequences of treatment decisions under uncertainty for patients at risk of life-threatening conditions. While highly accurate laboratory tests can inform such treatment decisions, the administration of these tests carry a significant cost for the patient and the health system [16]. These costs, together with the fact that frequent testing may be redundant and wasteful [7, 64], appropriately lead clinicians to refrain from constantly observing the patient state (i.e., ordering laboratory tests) [1, 11]. This action-contingent state observation setting can appropriately be modeled as an ACNO-MDP.

ACNO-MDPs for user-adaptive experiences. Applications on mobile phones and other personal devices can collect information on the user's status, such as location, motion, inter-user contact, and background noise, in order to adaptively suggest features that enhance the user's experience [57, 58]. While these sensors can in theory always be kept active, battery consumption due to constant sensing would make such applications less desirable to the user. Modeling this problem as an ACNO-MDP would enable one to learn a policy that balances observing the user state with capitalizing on opportunities to enhance the user's experience.

Our contributions Our main contributions include:

- Proposing a Probably Approximately Correct "Observe-then-Plan" algorithm for tabular ACNO-MDPs that fully observes while exploring, then employs POMDP planning, with the goal of learning a policy that can simultaneously decide (1) when to observe, and (2) how to act near-optimally in environments with state observations that are costly but optional.
- Providing a finite sample guarantee for the above algorithm in environments with discrete states and actions, such that the number of episodes exhibiting worse than near-optimal behavior is no larger than $O(\frac{S^5 A^2 H^5 {R_{\max}}^3 \iota^4}{\epsilon^3})$ with $\iota = O(\ln(SAH/\delta\epsilon))$ with high probability. This is significantly improved relative to prior work which tackles a more general class of POMDP problems (see Table 2).
- Developing an algorithm that extends these ideas to ACNO-MDPs with continuous state and action spaces. This includes the implementation of a novel belief state encoder that can be coupled with many modern deep RL methods to solve problems with observation costs.

2 Related work

Learning in POMDPs without a known model of the environment (i.e., with neither a model of state-to-state transition dynamics nor a model of observations conditioned on states) is significantly more difficult than learning in MDPs [39]. Thus while there is a rich literature on planning in POMDPs with a known environment model [3, 5, 8, 14, 26, 29, 30, 40, 43, 44, 47, 49, 51, 55, 63], there are fewer established approaches for reinforcement learning in POMDPs (POMDP-RL). One key difficulty in general POMDP-RL is that, without a known observation model, the agent can never know with certainty which states were actually visited. Approximating a transition model for use in subsequent planning thus becomes infeasible without additional assumptions on the observation model. Many approaches in POMDP-RL, such as the Bayes-Adaptive POMDP (BA-POMDP) [42, 46] and its variants [22, 23], assume access to an informative prior distribution over the observation model and maintain posterior distributions over the environment model parameters. These algorithms plan using belief distributions over states rather than the states themselves. More recent approaches like Deep Recurrent Q-Networks (DRQN) [13, 65] and Deep Variational Reinforcement Learning (DVRL) [33] use neural network-based sequence models to learn a condensed representation of the agent's action and observation history and apply modern RL algorithms (e.g., DQN [34], A2C [60]) on this learned representation. While these more recent approaches show promising empirical performance, they lack any theoretical performance guarantees.

Theoretical guarantees on a POMDP-RL algorithm's performance can be paramount when the algorithm is used to make safety-critical decisions, and especially when gathering samples under a suboptimal policy is costly. Though recent works have been able to provide theoretical guarantees

in surprisingly broad classes of POMDPs, these works either provide regret guarantees (not sample complexity guarantees) [15, 54], or make assumptions about the structure of the observation matrix which do not hold in ACNO-MDPs (e.g., that the observation matrix is full rank [12, 2] or its singular value is bounded away from 0 [18]). To the best of our knowledge, our work therefore provides sample complexity guarantees for a subclass of POMDPs in which no such prior guarantees existed.

Closely related to our work, Klima et al. [24] consider reinforcement learning in environments where the state is fully observed, but only intermittently. However, while Klima et al. [24] consider environments in which the agent cannot control state observation frequency, we address the problem of RL in environments where the agent can *decide* when to observe, albeit at a cost. In ACNO-MDPs with state observation costs, the agent must therefore learn to balance the benefit of observing (for the purposes of both exploration and planning) with the cost of obtaining that information.

3 Problem setting

Action-contingent noiselessly observable Markov decision processes (ACNO-MDPs). A discounted, finite-horizon ACNO-MDP is an environment defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p, \gamma, r, b, H \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces with cardinalities S and A, respectively. The action space $\mathcal{A}$ consists of tuples $\{\text{control actions}\} \times \{\text{observe, not observe}\}$. As in POMDPs, the observation space, $\mathcal{O}$, has cardinality O, and is related to $\mathcal{S}$ by the observation/emission function $p(o|s'a)$, which defines the probability of observing o if the agent has taken action a and is now in s'. In ACNO-MDPs, however, the observation space is specifically constrained to be $\mathcal{O} = \mathcal{S} \cup \{\text{missingness}\}$, where the "missingness" observation provides no information whatsoever about the underlying state. The "action-contingent noiselessly observable" aspect of ACNO-MDPs derives from the fact that $p(o|s', a_{\text{observe}}) = 1$ if and only if $o = s'$, while $p(o'|s', a_{\text{observe}}) = 0$ for all other observations $o' \neq s'$. Similarly, $p(o|s', a_{\text{not observe}}) = 1$ if and only if $o =$ "missingness". These facts represent a key difference between ACNO-MDPs and general POMDPs; leveraging knowledge of this observation model - specifically, that observations exactly match underlying states following a choice to observe - is critical for efficient learning in ACNO-MDPs.

While observations in ACNO-MDPs are deterministic, transition dynamics may be stochastic. The probability of transitioning to state s' after taking action a from state s is given by $p(s'|s, a)$. Because in ACNO-MDPs the agent faces state uncertainty if it chooses not to observe, we let b represent a belief distribution over possible states. We assume the initial belief state $b_0 = s_0$ is known (i.e., s_0 is always observed), but note that b_0 may be drawn from a probability distribution over states. We use a subscript t to indicate the time step within the episode, so that a_t and s_t denote the action a taken from state s at time step t. Letting H represent the episode length, we have $1 \leq t \leq H$. We use $n(s, a)$ to denote the number of times the state-action pair (s, a) is visited across episodes.

As in general POMDPs, we let $r(s, a) \in [0, R_{\text{max}}]$ represent the expected instantaneous reward obtained from executing action a in state s. Unlike POMDPs, however, ACNO-MDPs include observation costs in the reward function. Letting $c(s, a) \leq 0$ represent the cost of observation from state s and $\tilde{r}(s, a)$ represent the control-associated reward typically seen in POMDPs (e.g., the reward from reaching a goal state), the reward function for an ACNO-MDP is given as $r(s, a) = \tilde{r}(s, a) + c(s, a)$. For actions a in $\mathcal{A}$ that represent a choice to observe the state (indicated using subscript notation, i.e., a_{observe}), we have $c(s, a_{\text{observe}}) \leq 0$. For actions a in $\mathcal{A}$ representing a choice not to observe the state, we have $c(s, a_{\text{not observe}}) = 0$. Thus, for any action a, $r(s, a_{\text{observe}}) = r(s, a_{\text{not observe}}) + c(s, a_{\text{observe}})$ where $c(s, a) \leq 0$. For our analysis, we assume $0 \leq r \leq R_{\text{max}}$, which can be achieved for any ACNO-MDP by translating and scaling the reward. We apply a discount factor $0 < \gamma \leq 1$ to the reward. We assume stationary dynamics, observations, and rewards, but allow for nonstationary policies in light of the fixed horizon length. A t-step policy, $\pi_t(\cdot)$, is a function that maps states to distributions over actions. We let $V^{\pi_t}(s_0) = \mathbb{E}_{(s_i, a_i) \sim \pi_t} \sum_{i=0}^{t-1} \gamma^i r(s_i, a_i)$ represent the expected return obtained from starting in s_0 and following t-step policy, π_t. An optimal policy, π^*, is one for which $V^{\pi^*}(s) = \max_\pi V^\pi(s)$ for all states s in $\mathcal{S}$. The goal of ACNO-MDP learning is to find the policy π^* that maximizes $V^{\pi^*}(b_0)$, the expected return obtained from following π^* from initial belief distribution b_0. We evaluate our algorithms using the notion of sample complexity [21, 53], a measure of how much data is needed to learn a near-optimal policy. We say that an algorithm is Probably Approximately Correct (PAC) if its sample complexity is polynomial in relevant quantities (e.g., S, A, H) with high probability [53].

Table 1: Summary of algorithms considered

Framework	Algorithm name	Setting
Observe before planning	*Observe-then-Plan* (Ours)	Tabular
Observe while planning	ACNO-POMCP (Ours)	Tabular
Generic POMDP-RL	DRQN [13]	Tabular
MDP-RL	EULER-POMCP [62]	Tabular
Observe while planning	ACNO-A2C (Ours)	Continuous States/Actions
Generic POMDP-RL	DVRL-A2C [33]	Continuous States/Actions
MDP-RL	A2C [35, 60]	Continuous states/Actions

4 Algorithms

We propose two high-level frameworks for finding a near-optimal policy in ACNO-MDP settings: (1) **"Observe before planning"** (Algorithm 1), which initially spends a fixed number of timesteps consistently observing the state in order to learn accurate models of the transition dynamics and rewards, as if in a full MDP, then switches to computing a near-optimal POMDP policy using these models; and (2) **"Observe while planning"**, which incorporates the known ACNO-MDP observation structure (i.e., that observation emissions are identical to underlying states following a choice to observe) into an existing POMDP-RL algorithm. Our proposal for "Observe before planning" follows the structure of EEPORL [12], which devotes a fixed number of episodes to exploration in order to accurately estimate models of transition dynamics, rewards, and the function mapping states to observations. The key difference is that our "Observe before planning" framework only needs to learn models of the transition dynamics and reward function, since the observation model is known and every transition can be exactly identified if the agent chooses to observe. The motivation for "Observe while planning" is that any existing POMDP-RL algorithm can be simplified by leveraging the known ANCO-MDP observation function, which gives the true underlying state if the agent chooses to observe and an uninformative state otherwise. The purpose of both frameworks is to improve sample efficiency by leveraging the known observation structure in ACNO-MDP settings.

Algorithm 1 Observe before planning

Set dataset $D \leftarrow \emptyset$
for episode = 1 to N **do** ▷ N is # of episodes spent exploring (e.g., Thm1 gives one example of N)
    Reset to initial state s_0
    for step = 1 to H **do**
        Pay to observe every (s, a, s', r) and store into D
    end for
end for
Calculate maximum likelihood estimates of transitions $\hat{p}(s, a, s')$ and rewards $\hat{r}(s, a)$ using D
Return policy $\hat{\pi}$ from POMDP planning under $\hat{p}, \hat{r}$

In our experiments, we instantiate these frameworks into specific algorithms and compare them against MDP-RL methods, which learn an always-observing policy regardless of the cost of observation, as well as state-of-the-art POMDP-RL algorithms (e.g., DRQN [13] and DVRL [33]), which do not leverage any knowledge about the ACNO-MDP observation function. A summary of all algorithms used for comparison is shown in Table 1; training details for each model are provided in the Appendix. In our analysis, we provide sample complexity guarantees for learning an ϵ-optimal policy following the "Observe before planning" framework.

4.1 *Observe-then-Plan* algorithm for tabular ACNO-MDPs ("Observe before Planning")

For tabular ACNO-MDP domains, we provide the *Observe-then-Plan* algorithm as an instantiation of the "Observe before planning" framework. *Observe-then-Plan* decouples exploration from planning, following prior work on reward-free exploration [19]. In contrast to this prior work, which addresses non-stationary domains, we focus on domains with stationary dynamics and rewards in a finite episodic setting. The first key algorithmic step in *Observe-then-Plan* is to visit all relevant state-action pairs sufficiently many times, while observing at every time step, so as to learn accurate models

of the transition dynamics and rewards. For the purposes of our theoretical analysis, we employ EULER [62] in this exploration phase, as it provides state-of-the-art algorithm cumulative regret guarantees for episodic MDPs. We note, however, that any MDP-RL algorithm can be used for this step. The second key algorithmic step in *Observe-then-Plan* is to use the models of transitions and rewards estimated from exploration data for POMDP planning. While in our analysis we leverage algorithms that guarantee ϵ-optimal performance in POMDP planning [46, 10], these approaches are often impractical for any reasonably-sized POMDP. In our experiments we therefore use POMCP, a popular and fast POMDP planning algorithm [49], for the planning phase.

4.2 ACNO-POMCP/ACNO-A2C algorithms ("Observe while Planning")

Although our theoretical guarantee (Theorem 1) requires the agent to spend a fixed number of episodes exploring while observing at every time step, this extensive exploration and observation may be unnecessary for obtaining near-optimal performance in practice. This motivates our second framework "Observe while planning," under which we incorporate the known observation structure into an existing state-of-the-art POMDP-RL method.

For tabular settings, we instantiate the "Observe while planning" framework with the ACNO-POMCP algorithm. ACNO-POMCP simultaneously (1) uses online POMCP to plan optimal trajectories based on simulated rollouts, and (2) updates the approximate reward and transition models used in these simulations every time a new sample (s, a, s', r) is observed from the true environment. Note the observation function does not need to be learned because the algorithm leverages the fact that $p(o|a_{\text{observe}}, s') = 1$ if and only if $o = s'$ and $p(o|a_{\text{not observe}}, s') = 1$ if and only if $o =$ "missingness". Thus model estimation in ACNO-POMCP reduces to MDP model estimation, but because the agent can choose when to observe the algorithm does not provide a guarantee on how frequently these observations are made.

For ACNO-MDPs with continuous state spaces, we instantiate the "Observe while Planning" framework with the ACNO-A2C algorithm. ACNO-A2C incorporates the known observation structure into a belief encoder [33]. This belief encoder maintains a set of (weighted) belief particles, each of which represents an encoding of the agent's potential state/action history. Whenever the agent chooses to observe, (1) a loss is calculated for each particle in proportion to the negative log-likelihood of visiting the observed state given the particle's (encoded) history, and (2) each particle's history is updated with the observed state. When the agent chooses not to observe, (1) a state is sampled for each particle from the belief encoder model conditioned on that particle's encoded history; (2) each particle is then assigned a weight in proportion to the sampled state's likelihood under that history encoding and the learned belief encoding model; and (3) each particle's history is updated with its sampled state. Our belief encoder (Algorithm 2), which adapts the DVRL belief encoder [33] to the ACNO-MDP setting, can be used with any deep RL method using the belief state encoding. We instantiate it with the actor-critic method A2C [35, 60] (thus the name ACNO-A2C).

Algorithm 2 ACNO belief encoder

Input Previous belief set $\hat{b}_{t-1}$, observation o_t, action a_{t-1}.
Output $\hat{b}_t$, and belief encoder loss $\mathcal{L}_t^{\text{enc}}$.

Unpack $w_{t-1}^{1:K}, z_{t-1}^{1:K}, h_{t-1}^{1:K}, \hat{h}_{t-1} \leftarrow \hat{b}_{t-1}$
for every particle $k = 1, \ldots, K$ **do**
    Resample previous context $h_{t-1}^k \sim h_{t-1}^{1:K}$ based on weights
    if $a_{t-1} =$ observe **then**
        $z_t^k = o_t, w_t^k = \frac{1}{K}$
        $\mathcal{L}_t^{\text{enc}} \leftarrow \mathcal{L}_t^{\text{enc}} - \log p_\theta(o_t|h_{t-1}^k, a_{t-1})$
    else
        Sample $z_t^k \sim p_\theta(z_t^k|h_{t-1}^k, a_{t-1})$
        $w_t^k \leftarrow p_\theta(z_t^k|h_{t-1}^k, a_{t-1})$
    end if
    $h_t^k \leftarrow \text{GRU}(h_{t-1}^k, z_t^k, o_t, a_{t-1})$
end for
$\hat{h}_t \leftarrow \text{GRU}(\text{Concat}(w_t^k, z_t^k, h_t^k)_{k=1}^K)$
Pack $\hat{b}_t \leftarrow (w_t^{1:K}, z_t^{1:K}, h_t^{1:K}, \hat{h}_t)$ and include $\mathcal{L}_t^{\text{enc}}$ if $a_{t-1} =$ observe.

Table 2: Sample complexity comparisons

Algorithm	Setting	Transition	Sample Complexity
OOM-UCB [17]	POMDP	non-stationary	$\tilde{O}\left(\frac{H^4 S^6 A^4 O^3}{\sigma_{\min}(\mathbb{O})^8 \epsilon^2}\right)$
EEPORL [12]	POMDP	stationary	$\tilde{O}\left(\frac{H^4 V_{\max}^2 A^{12} R^4 O^4 S^{12}}{C_{d,d,d}(\frac{\delta}{3}^2)\underline{\sigma}_a(T_a)^6 \underline{\sigma}_a(R_a)^8 \underline{\sigma}_a(O_a)^8 \epsilon^2}\right)$
Observe-then-Plan	ACNO-MDP	stationary	$\tilde{O}\left(\frac{H^5 S^5 A^2 R_{\max}^3}{\epsilon^3}\right)$

5 Analysis

Observe-then-Plan reduces state-of-the-art PAC POMDP-RL sample complexity bounds (OOM-UCB [17], Table 2) by a factor of A^2SO^3 in ACNO-MDP settings, while increasing these bounds by a factor of $\frac{H}{\epsilon}$ and logarithmic quantities. We note, however, that prior state-of-the-art PAC POMDP-RL algorithms make assumptions on the structure of the observation matrix that do not hold in ACNO-MDPs, thus limiting the utility of direct comparisons. Specifically, when an agent chooses not to observe in the tabular ACNO-MDP setting, the corresponding observation matrix $\mathbb{O} \in \mathbb{R}^{(S+1)\times S}$ has a single row with values $1/S$ and all other rows are zero. This leads to rank deficiency in the observation matrix, violating assumptions made in Guo et al. [12], and $\sigma_{\min}(\mathbb{O}) = 0$, violating the assumptions made in Jin et al. [17]. Thus, to the best of our knowledge, our results provide sample complexity guarantees for a class of POMDPs not covered by prior POMDP-RL work. These guarantees hold for ACNO-MDPs with arbitrary state dynamics and hard-to-reach states.

Theorem 1. *Let $\mathcal{M}$ be any ACNO-MDP with observation cost $c(s, a_{\text{observe}}) < 0$ for all $(s, a) \in S \times A$, and reward bounded between $[0, R_{\max}]$. Let b_0 be the initial belief, and let ϵ and δ be two positive real numbers. Following* Observe-then-Plan *will achieve an expected episodic reward of $V(b_0) \geq V^*(b_0) - \epsilon$ after a number of episodes that is bounded by $O(\frac{S^5 A^2 H^5 R_{\max}^3 \iota^4}{\epsilon^3})$, where $\iota = O(\ln(SAH/\delta\epsilon))$, with probability at least $1 - \delta$.*

Proof sketch. (Full proofs provided in the Appendix). First, we note that prior work on POMDP-RL provides guarantees on expected performance, i.e. $V(b_0) \geq V^*(b_0) - \epsilon$, provided that value function estimates are sufficiently close to the true value function [12]. With sufficiently accurate estimates of the ACNO-MDP transition and rewards models, we can bound the error of the value function estimates computed under those models (following the EEPORL analysis) [10, 12, 45]. Prior work on Model-based Interval Estimation [52, 59] allows us to bound model estimate errors (i.e., $||\hat{p} - p||_1$ and $|\hat{r} - r|$) in terms of $n(s, a)$. Using this, we can find the number of times each state-action pair must be visited in order to guarantee sufficiently accurate ACNO-MDP model estimates. Call this number m. Reward-free exploration [19] with EULER [62] guarantees that, with high probability, the agent will visit each state-action pair at least m times after a certain number of episodes, N. We show that this number N is $O(\frac{S^5 A^2 H^5 R_{\max}^3 \iota^4}{\epsilon^3})$.

Only the exploration phase of *Observe-then-Plan* contributes to its sample complexity, as planning requires no additional learning. In generic POMDP settings where the true underlying states are inaccessible, bounding the estimated model errors is difficult because the number of times each state-action pair is visited cannot be measured. In ACNO-MDP settings, transition and reward models can be estimated via simple MLE if the agent is willing to initially pay the cost of observation.

6 Experimental results

In order to analyze the empirical performance of our proposed methods in both tabular and continuous state-action spaces, we evaluate the algorithms from Table 1 in (1) a tabular ACNO-MDP environment designed to simulate patient care in the Intensive Care Unit; (2) a continuous-state, discrete-action "Cart Pole" environment; and (3) a continuous-state, continuous-action "Mountain Hike" environment.

6.1 Algorithms for tabular ACNO-MDPs

Sepsis simulator Our first environment simulates a Sepsis patient and the effects of several common treatments. Adapted from Oberst and Sontag [38], our "Sepsis simulator" defines the state space as

{heart rate, blood pressure, oxygen concentration, glucose level}, each of which can take on several discretized state values (so in total, tabular $S = 720$). The action space is defined as a combination of binary treatment options consisting of {antibiotics, vasopressors, ventillation}, each of which can either be "on" or "off". We augment the action space by coupling each action with options of observing and not observing. A reward of 0 is given if the patient reaches a terminal "death" state, 1 if the patient reaches a terminal "discharge" state, and 0.25 otherwise. We provide results for each algorithm under 2 different fixed observation costs, -0.1 and -0.05.

Algorithms We compare four different methods for learning an optimal policy in the Sepsis simulator.

Observe-then-Plan ("Observe before planning") is our originally proposed *Observe-then-Plan* algorithm (Section 4.1) with a reduced number of episodes (compared to the proposed $H^5S^5A^2/\epsilon^3$ episodes) spent on exploration, in order to make the algorithm computationally tractable.

ACNO-POMCP ("Observe while planning") is an adaptation of the POMCP online planning algorithm to the learning setting based on the open-sourced POMCP codebase [9]. At each time step, *ACNO-POMCP* uses Monte Carlo tree search to select actions (including observing and non-observing actions) using an approximated model of the environment and an ε-greedy exploration strategy, with a decaying ε. Whenever *ACNO-POMCP* chooses to observe the state twice in a row, the observed (s, a, s', r) tuple is used to update the parameters of its approximate transition and rewards models. *ACNO-POMCP* implicitly uses knowledge of the ACNO-MDP observation structure by treating emitted observations as ground truth state observations and updating its reward and transition estimates over states.

DRQN (Generic POMDP-RL) leverages an LSTM layer to encode sequences of one-hot-encoded observations into a single hidden state encoding [13][1]. The algorithm learns a mapping from this hidden state to Q estimates, which are used for action selection under a decaying ε-greedy strategy.

EULER-POMCP (Always observing MDP-RL) chooses to observe at every time step, essentially treating the ACNO-MDP as an MDP with constant observation costs. Initially, EULER-POMCP uses EULER [62] to strategically explore the state-action space. After a fixed number of "exploration' episodes, EULER-POMCP computes Value Iteration under maximum likelihood estimates of the transition dynamics and rewards models to learn an optimal five-step nonstationary policy.

We evaluate the policy obtained from each method with 50 rollouts in the true environment and report the obtained discounted returns including the observation costs (Table 3).

Results We ran each of the above methods for $2,000$ episodes on the Sepsis simulator domain. The "exploration" phases of the *Observe-then-Plan* and *EULER-POMCP* algorithms were fixed to be $1,000$ episodes, at which point learning stopped. *ACNO-POMCP* and *DRQN* learned continuously for all $2,000$ episodes. Learning curves for each algorithm are shown in Figure 1 and performance of the final policies (i.e., after $2,000$ episodes) for each algorithm are shown in Table 3.

Figure 1 shows that the algorithms without any explicit "exploration" phase (i.e., *DRQN* and *ACNO-POMCP*) achieved higher returns during the first $1,000$ episodes compared to algorithms that used the first $1,000$ episodes to actively explore the state-action space while always observing (i.e., *Observe-then-Plan*, *EULER-POMCP*). As expected, rewards under both the *Observe-then-Plan* and *EULER-POMCP* algorithms improved dramatically after switching from "exploration" to "planning", but while Observe-then-Plan was able to achieve an optimal policy by balancing the tradeoff between observing and controlling, *EULER-POMCP* continued to observe at every time step and suffered in performance as a result. We note that, in the Sepsis simulator, any policy that prevents the patient from dying or being discharged for all 5 steps achieves a discounted return of 0.693. *Observe-then-Plan* is the only algorithm that outperforms this neutral policy in expectation.

6.2 Algorithms for ACNO-MDPs with continuous state-action spaces

Explicit reward-free exploration is difficult to do in continuous state and action spaces where the number of times an agent visits a unique state-action pair cannot be measured effectively. Our *ACNO-A2C* algorithm instantiates our second framework of "Observe-while-planning" to simultaneously learn an accurate model of the environment while maximizing discounted episode returns. Our

[1]For implementation, we modified the open-sourced code at `https://github.com/Bigpig4396/PyTorch-Deep-Recurrent-Q-Learning-DRQN`. The link to our codebase as well as detailed pseudocodes are included in the Appendix.

Table 3: Average discounted returns (mean $\pm$ 1 ste) on Sepsis. Standard errors represent the standard error of the mean return obtained from 3 separate seeds, each with rewards averaged across 50 simulated rollouts at the end of 2k training episodes using greedy action selection.

Observation Cost	*Observe-then-Plan* (Observe before planning)	ACNO-POMCP (Observe while planning)	DRQN (Generic POMDP-RL)	EULER-POMCP (MDP-RL)
-0.1	**0.754** $\pm$ **0.011**	0.602 $\pm$ 0.029	0.593 $\pm$ 0.079	0.492 $\pm$ 0.011
-0.05	**0.740** $\pm$ **0.017**	0.625 $\pm$ 0.024	0.593 $\pm$ 0.036	0.638 $\pm$ 0.025

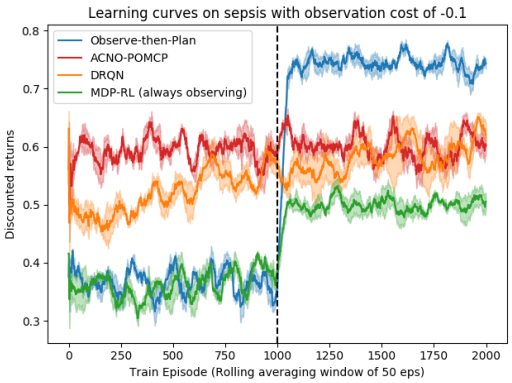

Figure 1: Sepsis learning curves under the observation cost -0.1, with a rolling average window of size 50. Shaded regions indicate $\pm$ 1 standard error over 3 simulated runs. The dotted line at 1000 indicates the episode at which *Observe-then-Plan* and EULER-POMCP switch to planning. DRQN and ACNO-POMCP continue learning. The x-axis shows the number of training episodes and the y-axis shows true observed episode returns from the environment.

results suggest that our proposed belief state encoder based on "Observe-while-planning" empirically outperforms state-of-the-art algorithms without having to explore all of the state and action spaces.

Experimental domains

Modified CartPole simulator. We test on OpenAI gym `CartPole-v0` [4] with the following modifications: (1) the action space is doubled in size to {push left, push right} $\times$ {observe, not observe}; (2) a cost of -0.4 is applied to every reward $r(s, a_{\text{observe}})$ in which the agent chooses to fully observe the state; and (3) if a state is not observed, then the environment returns all zeros for the observation.

Mountain Hike simulator. We also test on a two-dimensional Mountain Hike environment, as described and implemented by Maximilian et al. [33]. The goal of this task is to follow a high reward path along a "mountain ridge". The state space is defined by the current position, (x, y), and the action a is defined as any possible vector $(\triangle x, \triangle y) \in \mathbb{R}^2$ with bounded ℓ_1-norm. The agent starts from sampled coordinates $(x_0, y_0) \sim N(-8.5 I_{2\times 2}, I_{2\times 2})$. Observations returned are exactly the underlying state, except when the agent chooses not to observe in which case observations are a vector of zeros. Transitions are defined as $(x', y') = (x, y) + (\triangle x, \triangle y) + \varepsilon, \varepsilon \sim \mathcal{N}(0, 0.025 I_{2\times 2})$. The reward is given by $\gamma(x, y) - 0.01||a||_1 + c_{obs}(a)$, where $\gamma(x, y)$ is the reward associated with the agent's proximity to the "mountain ridge" and $c_{obs}(a)$ is the cost of fully observing the state, such that $c_{obs}(a_{\text{observe}}) = -0.5$ if the agent chooses to observe and $c_{obs}(a_{\text{not observe}}) = 0$ otherwise.

Algorithms

ACNO-A2C ("Observe while Planning") is the algorithm instantiating the belief encoder in Algorithm 2 with A2C, a well-known actor-critic reinforcement learning method [35, 60].

DVRL (Generic POMDP-RL) uses an implementation of the Deep Variational Reinforcement Learning algorithm by Maximilian et al. [33]. As a state-of-the-art POMDP-RL approach for general POMDPs, this algorithm does not leverage the special observation model given in ACNO-MDPs and must learn both the transition dynamics and the observation model online, while simultaneously attempting to maximize overall reward.

A2C (Always observing MDP-RL) applies the actor-critic method [35, 60] directly to the state, assuming the agent always chooses to observe the underlying state and incurs the associated observation cost. We use the A2C implementation by Maximilian et al. [33] but modified to be defined over states.

Both the *ACNO-A2C* and *DVRL* algorithms use an encoder that returns an array of aggregated belief particles. This set of belief particles is then used for learning an A2C policy and value function.

Results Figure 2 shows the learning curves for our proposed algorithm (ACNO-A2C), the Deep Variational Reinforcement Learning algorithm (DVRL), and the always-observing A2C algorithm. Results are averaged over 5 random seeds. Shaded regions show $\pm$ 1 standard deviations over the 5 runs. Given an equal number of training steps, our ACNO-A2C algorithm achieved a higher-reward final policy, in expectation, and found this high-reward policy in almost half the time that it took for the DVRL algorithm to find its highest-reward policy. Running standard A2C while always observing achieved high returns faster than ACNO-A2C (*Mountain Hike*) or about as fast as ACNO-A2C (`CartPole-v0`), but achieved an inferior final policy relative to ACNO-A2C.

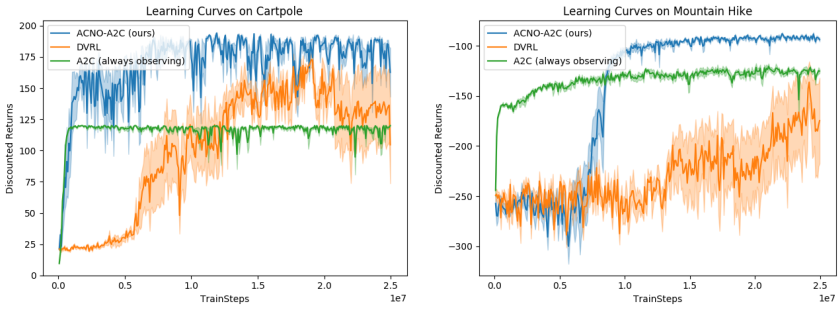

Figure 2: Left: CartPole-v0 with observation cost 0.4. Right: Mountain Hike with cost 0.5.

Our findings suggest that leveraging information about the structure of the observation model can lead to improved learning efficiency (in terms of sample complexity) and higher-reward policies compared to state-of-the-art generic POMDP-RL algorithms which make minimal assumptions about the structure of the observation model.

7 Conclusion

We defined a special class of POMDPs, ACNO-MDPs, that frequently arise in a number of interesting and important real-world problem domains. We illustrated how assumptions made by many existing PAC POMDP-RL algorithms regarding the observation function are violated in ACNO-MDPs, but we proposed and analysed a PAC RL *Observe-then-Plan* algorithm for efficient learning in these settings.. Building on this idea, we proposed a method for efficiently encoding belief states in ACNO-MDPs, and a meta-algorithm that can couple this belief state representation with modern deep RL algorithms in order to learn policies that balance the cost of observing with the benefits of observing for exploration and planning. We found that our proposed algorithm learns faster and finds better policies compared to existing state-of-the-art POMDP-RL approaches targeted towards more general POMDP problems.

We note that ACNO-MDPs assume the observation cost is quantifiable and that the act of observing does not affect the underlying dynamics of the system. Our proposed frameworks might yield competitive results in settings where these conditions are not rigorously satisfied, but our theoretical guarantees from *Observe-then-Plan* are limited to such tabular settings. While we considered the online setting exclusively in this work, we hope to extend these ideas to the offline/batch RL setting. In many problems appropriately modeled as ACNO-MDPs, online learning may be infeasible and/or unethical (e.g., deploying our ACNO-POMCP algorithm in the ICU to guide Sepsis management could raise ethical concerns). Leveraging our ideas to learn optimal policies using data collected from suboptimal behavior policies could help in achieving practical policies that balance real-world costs of observing with other competing environment objectives, but these policies should be rigorously inspected and evaluated before real-world deployment.

References

[1] Denis Agniel, Isaac S Kohane, and Griffin M Weber. Biases in electronic health record data due to processes within the healthcare system: retrospective observational study. *Bmj*, 361, 2018.

[2] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning of pomdps using spectral methods. In *Conference on Learning Theory*, pages 193–256. PMLR, 2016.

[3] Haoyu Bai, David Hsu, and Wee Sun Lee. Integrated perception and planning in the continuous space: A pomdp approach. *The International Journal of Robotics Research*, 33(9):1288–1302, 2014.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Anthony R Cassandra, Michael L Littman, and Nevin Lianwen Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 54–61, 1997.

[6] Antonio Coronato, Muddasar Naeem, Giuseppe De Pietro, and Giovanni Paragliola. Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine*, 109:101964, 2020.

[7] Kevin P Eaton, Kathryn Levy, Christine Soong, Amit K Pahwa, Christopher Petrilli, Justin B Ziemba, Hyung J Cho, Rodrigo Alban, Jaime F Blanck, and Andrew S Parsons. Evidence-based guidelines to eliminate repetitive laboratory testing. *JAMA internal medicine*, 177(12): 1833–1839, 2017.

[8] Maxim Egorov, Zachary N Sunberg, Edward Balaban, Tim A Wheeler, Jayesh K Gupta, and Mykel J Kochenderfer. Pomdps. jl: A framework for sequential decision making under uncertainty. *The Journal of Machine Learning Research*, 18(1):831–835, 2017.

[9] P Emami, AJ Hamlet, and C Crane. Pomdpy: An extensible framework for implementing pomdps in python. 2015. URL `https://github.com/pemami4911/POMDPy`.

[10] Mahdi Milani Fard, Joelle Pineau, and Peng Sun. A variance analysis for pomdp policy evaluation. In *AAAI*, pages 1056–1061, 2008.

[11] Scott L Fleming, Kuhan Jeyapragasan, Tony Duan, Daisy Ding, Saurabh Gombar, Nigam Shah, and Emma Brunskill. Missingness as stability: Understanding the structure of missingness in longitudinal ehr data and its impact on reinforcement learning in healthcare. *arXiv preprint arXiv:1911.07084*, 2019.

[12] Zhaohan Guo, Shayan Doroudi, and Emma Brunskill. A PAC RL algorithm for episodic pomdps. *CoRR*, abs/1605.08062, 2016. URL `http://arxiv.org/abs/1605.08062`.

[13] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[14] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[15] Mehdi Jafarnia-Jahromi, Rahul Jain, and Ashutosh Nayyar. Online learning for unknown partially observable mdps. *arXiv preprint arXiv:2102.12661*, 2021.

[16] Ashish K Jha, David C Chan, Abigail B Ridgway, Calvin Franz, and David W Bates. Improving safety and eliminating redundant tests: cutting costs in us hospitals. *Health affairs*, 28(5): 1475–1484, 2009.

[17] Chi Jin, Sham Kakade, Akshay Krishnamurthy, and Qinghua Liu. Sample-efficient reinforcement learning of undercomplete pomdps. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18530–18539. Curran Associates, Inc., 2020.

[18] Chi Jin, Sham M Kakade, Akshay Krishnamurthy, and Qinghua Liu. Sample-efficient reinforcement learning of undercomplete pomdps. *arXiv preprint arXiv:2006.12484*, 2020.

[19] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. 119:4870–4879, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/jin20d.html`.

[20] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[21] S. Kakade. On the sample complexity of reinforcement learning. 2003.

[22] Sammie Katt, Frans A Oliehoek, and Christopher Amato. Learning in pomdps with monte carlo tree search. In *International Conference on Machine Learning*, pages 1819–1827. PMLR, 2017.

[23] Sammie Katt, Frans Oliehoek, and Christopher Amato. Bayesian reinforcement learning in factored pomdps. *arXiv preprint arXiv:1811.05612*, 2018.

[24] Richard Klima, Karl Tuyls, and Frans A Oliehoek. Model-based reinforcement learning under periodical observability. In *2018 AAAI Spring Symposium Series*, 2018.

[25] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

[26] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Citeseer, 2008.

[27] Lei Lei, Yue Tan, Kan Zheng, Shiwen Liu, Kuan Zhang, and Xuemin Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1722–1760, 2020.

[28] Yuxi Li. Deep reinforcement learning. *arXiv preprint arXiv:1810.06339*, 2018.

[29] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995*, pages 362–370. Elsevier, 1995.

[30] William S Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations research*, 39(1):162–175, 1991.

[31] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3133–3174, 2019.

[32] Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, 29(6):2063–2079, 2018.

[33] Igl Maximilian, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *CoRR*, abs/1806.02426, 2018. URL `http://arxiv.org/abs/1806.02426,https://github.com/maximilianigl/DVRL`.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[35] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[36] Rui Nian, Jinfeng Liu, and Biao Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, page 106886, 2020.

[37] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[38] Michael Oberst and David Sontag. Counterfactual off-policy evaluation with Gumbel-max structural causal models. 97:4881–4890, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/oberst19a.html`,`https://github.com/clinicalml/gumbel-max-scm`.

[39] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[40] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032. Citeseer, 2003.

[41] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[42] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive pomdps. In *NIPS*, pages 1225–1232, 2007.

[43] Stéphane Ross, Brahim Chaib-Draa, et al. Aems: An anytime online search algorithm for approximate policy refinement in large pomdps. In *IJCAI*, pages 2592–2598, 2007.

[44] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[45] Stephane Ross, Masoumeh Izadi, Mark Mercer, and David Buckeridge. Sensitivity analysis of pomdp value functions. In *2009 International Conference on Machine Learning and Applications*, pages 317–323. IEEE, 2009.

[46] Stéphane Ross, Joelle Pineau, Brahim Chaib-draa, and Pierre Kreitmann. A bayesian approach for learning and planning in partially observable markov decision processes. *Journal of Machine Learning Research*, 12(5), 2011.

[47] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[48] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

[49] David Silver and Joel Veness. Monte-carlo planning in large pomdps. Neural Information Processing Systems, 2010.

[50] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[51] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *NIPS*, volume 13, pages 1772–1780, 2013.

[52] Alexander Strehl and Michael Littman. A theoretical analysis of model-based interval estimation. page 856–863, 2005. doi: 10.1145/1102351.1102459. URL `https://doi.org/10.1145/1102351.1102459`.

[53] Alexander Strehl, Lihong Li, and Michael Littman. Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(84):2413–2444, 2009. URL `http://jmlr.org/papers/v10/strehl09a.html`.

[54] Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *arXiv preprint arXiv:2010.08843*, 2020.

[55] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 2018.

[56] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[57] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, page 179–192, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585666. doi: 10.1145/1555816.1555835. URL `https://doi.org/10.1145/1555816.1555835`.

[58] Yi Wang, Bhaskar Krishnamachari, Qing Zhao, and Murali Annavaram. Markov-optimal sensing policy for user state estimation in mobile devices. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, page 268–278, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589886. doi: 10.1145/1791212.1791244. URL `https://doi.org/10.1145/1791212.1791244`.

[59] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marco Weinberger. Inequalities for the l1 deviation of the empirical distribution. 2003.

[60] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *arXiv preprint arXiv:1708.05144*, 2017.

[61] Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.

[62] Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019*, abs/1901.00210, 2019. URL `http://arxiv.org/abs/1901.00210`.

[63] Nevin L Zhang and Wenju Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical report, Technical Report HKUST-CS96-31, Hong Kong University of Science and Technology, 1996.

[64] Ming Zhi, Eric L Ding, Jesse Theisen-Toupal, Julia Whelan, and Ramy Arnaout. The landscape of inappropriate laboratory testing: a 15-year meta-analysis. *PloS one*, 8(11):e78962, 2013.

[65] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] All our algorithms and experimental domains are open-source (e.g., POMDPy [9], DVRL/mountain hike [33], Sepsis simulator [38], CartPole from OpenAI-Gym [4], DRQN is modified from `https://github.com/Bigpig4396/PyTorch-Deep-Recurrent-Q-Learning-DRQN`). The link to our codebase is included in the Appendix.
- Did you include the license to the code and datasets? [Yes] See above.

- Did you include the license to the code and datasets? [Yes] All algorithms and experimental domains are open-source. Links are cited and our codebase will be released/linked after review.

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Limitations are discussed throughout (e.g., sample complexity only for tabular domains), in addition to the conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In our conclusion, we caution against using online algorithms to solve high-stake problems like Sepsis, which would be infeasible and/or unethical.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] These are included in the Appendix A. Proofs
   (b) Did you include complete proofs of all theoretical results? [Yes] These are included in the Appendix A. Proofs
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A GitHub repo with code and instructions needed to reproduce the main experimental results is provided in the Appendix, along with more detailed pseudocodes, (access will be granted selectively until acceptance, at which point the entire repo will be released) and DRQN is based on `https://github.com/Bigpig4396/PyTorch-Deep-Recurrent-Q-Learning-DRQN`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] These are included in the Appendix B. for Sepsis, CartPole, and Mountain Hike experiments, and associated links are cited.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results are reported with mean and 1 standard error over multiple runs with different random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Computation was performed on internal university servers. Wall clock-time for the experiments will be included in the Appendix B along with other training details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Algorithms we used in our work, including DVRL, DRQN, POMCP, etc., and environments, including Sepsis simulator, OpenAI CartPole, and DVRL mountain hike, are all cited in our manuscript.
   (b) Did you mention the license of the assets? [N/A] All considered algorithms and experimental domains are open-source (e.g., OpenAI-Gym, Sepsis) and require no additional license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code will be provided via a publicly accessible GitHub link, as described above.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We did not use any data from human subjects.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We used simulated data only.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No human subjects data was used in this project.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] See above.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] See above.