CONVERTING POMDPS INTO MDPS USING HISTORY REPRESENTATION

Khanh Nguyen
Department of Computer Science
University of Maryland, College Park
kxnguyen@umd.edu

1 INTRODUCTION

Partially-observed Markov decision process (POMDP) is a powerful framework for modeling sequential decision-making with incomplete information about the world. Defining standard utility functions like the V and Q functions is essential for developing reinforcement learning (RL) solutions. Nevertheless, this task is not trivial in the POMDP setting: we cannot simply take the functions that are constructed for MDPs and use them for POMDPs. Specifically, MDP utility functions take the full states as input (i.e. $V(s)$ and $Q(s, a)$). They are not useful for a POMDP agent because it cannot observe the full states. For POMDPs, ideally, we want to define these functions over histories of observations and actions, which the agent can observe. However, the number of possible histories grows exponentially with the trajectory length, making it impractical to reliably estimate history-based utility functions.

A common solution is to find a *compact representation* of the history. In early work on POMDP, Kaelbling et al. (1998) propose using belief states as the representation. A belief state is the distribution over states given a history. It is shown that belief states are a sufficient statistic of the history. Kaelbling et al. (1998) show that it is possible to convert an POMDP to an MDP that considers the belief states as its full states, making it possible to define utility functions and construct RL algorithms. Unfortunately, the dimension of belief states scales linearly with the number of states. Hence, They are not suitable for environments that have large state spaces.

More recently, with the resurgence of deep learning, researchers use recurrent neural networks to learn compact representations of history (Hausknecht and Stone, 2015). They apply algorithms designed for MDPs to POMDPs, treating the history representations as the input states of the utility functions. Conversion from POMDPs to MDPs using history representations is so effective and seamless that it is common to see a theory-practice gap in deep RL papers, where an algorithm is theoretically formulated in the MDP setting but is empirically evaluated on POMDP tasks without any justifications on why it would work in the latter setting (e.g., Mnih et al. (2013); Schulman et al. (2017); Chevalier-Boisvert et al. (2019); Wang et al. (2019)). Understanding this gap is important for extending the POMDP setting and developing more effective history representations.

This document provides a justification for the conversion from POMDPs to MDPs using history representations. Specifically, it provides answers for two questions:

1. *What characteristics must a history representation have to enable a valid conversion?* In the literature, it is often said that the representation should be a "sufficient statistic" of the history, but what does that mean formally? Section 2 introduces a mathematical definition of this notion.
2. *What MDP is a POMDP converted into?* While implementation-wise the conversion seems to simply treat the history representations as MDP states, it actually transforms a POMDP into an MDP whose transition dynamics and reward function are very different from those of the original POMDP. Section 3 presents a general conversion that works for any sufficient-statistic history representation.

Disclaimer: This document only serves educational purposes. It was written mainly for educating myself about POMDPs, and for sharing knowledge. These results may have been known to RL experts but I was not able to find a document where they were written down.

2 HISTORY REPRESENTATION

Environment. We consider a standard POMDP environment $(\mathcal{S}, \mathcal{A}, T, r, \Omega, O)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, state-transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$,[1] reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, observation space Ω, and a perception model $O : \mathcal{S} \times \mathcal{A} \to \Delta(\Omega)$.

The main difference between a POMDP and an MDP is that at any time step t, the agent never sees the (full) state $s_t \in \mathcal{S}$ of the environment. Instead, it perceives the environment through an observation $o_t \in \Omega$. Based on this observation, it may take an action $a_t \in \mathcal{A}$, and transitions to a new state $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, which again it does not see, and receives a reward $r(s_t, a_t, s_{t+1})$ for taking the action. In the new state s_{t+1}, it perceives a new observation $o_{t+1} \sim O(\cdot \mid s_{t+1}, a_t)$.

History Representation. Let $\tau_t = (o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t)$ be a history of observations and actions taken by the agent up to time step t. The space of all possible t-step histories grows exponentially with respect to t. Hence, it is not practical to define utility functions that take these raw histories as input.

Instead, we compute a representation $h_t = \Phi(\tau_t) \in \mathcal{H}$ of the history, where Φ is some feature extractor. An example of a history representation is a belief state b_t (Kaelbling et al., 1998), which specifies the probability of the agent being in each state given a history, i.e. $b_t(s) = P(s_t = s \mid \tau_t)$. A belief state can be viewed as an "average state": in a POMDP, because we are not sure which state the agent is in, we imagine that the agent is in *all* states at the same time and assign an importance weight to each state. Kaelbling et al. (1998) state that

> "*Furthermore, they [belief states] comprise a* ***sufficient statistic*** *for the past history and initial belief state of the agent: given the agent's current belief state (properly computed), no additional data about its past actions or observations would supply any further information about the current state of the world.*"

This sentence entails the following condition

$$P(s_t = s \mid \tau_t) = P(s_t = s \mid b_t) \quad \forall s \in \mathcal{S} \tag{1}$$

which the belief state obviously satisfies because $b_t(s) = P(s_t = s \mid \tau_t)$ by definition.

Unfortunately, belief states are not a compact representation because their dimension grows linearly with the number of states. For applications where the state space is large (e.g. a continuous state space), we want to learn a more compact representation of the history.

Inspired by the sufficient statistic condition of belief states, I define the following condition for a history representation.

Definition 2.1 (Sufficient history representation)**.** A representation h_t is said to be a *sufficient representation* of a history τ_t iff

$$b_t(s) = P(s_t = s \mid \tau_t) = P(s_t = s \mid h_t) \quad \forall s \in \mathcal{S} \tag{2}$$

Updating history representation. Kaelbling et al. (1998) show that a belief state can be computed recursively

$$\begin{aligned}
b_{t+1}(s_{t+1}) &= P(s_{t+1} \mid \tau_{t+1}) && (3)\\
&= P(s_{t+1} \mid \tau_t, a_t, o_{t+1}) && (4)\\
&\propto P(o_{t+1}, s_{t+1} \mid a_t, \tau_t) && (5)\\
&= P(o_{t+1} \mid s_{t+1}, a_t) P(s_{t+1} \mid a_t, \tau_t) && (6)\\
&= P(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} P(s_{t+1} \mid a_t, \tau_t, s_t) P(s_t \mid a_t, \tau_t) && (7)\\
&= P(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} P(s_{t+1} \mid a_t, \tau_t, s_t) P(s_t \mid a_t, \tau_t) && (8)\\
&= P(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} P(s_{t+1} \mid a_t, s_t) P(s_t \mid \tau_t) && (9)
\end{aligned}$$

[1] $\Delta(X)$ denotes the set of all probability distributions over X.

If h_t is a sufficient representation of τ_t, then the computation of b_{t+1} depends only on h_t, a_t, and o_{t+1}. This motivates computing h_{t+1} (the representation of b_{t+1}) recursively: $h_{t+1} = f(h_t, a_t, o_{t+1})$, as in a recurrent neural network. Here, I call f a *representation composer*.

An interesting question to ask is what condition(s) f should satisfy to guarantee that h_{t+1} is a sufficient representation of τ_{t+1}. I define the notion of a *sufficient-statistic composer*.

Definition 2.2 (Sufficient-statistic composer)**.** A function $f : \mathcal{H} \times \mathcal{A} \times \Omega \to \mathcal{H}$ is said to be a sufficient-statistic composer iff

$$P(s \mid f(h, a, o)) = P(s \mid h, a, o) \quad \forall h \in \mathcal{H}, o \in \Omega, a \in \mathcal{A} \tag{10}$$

It is easy to show that if f is a sufficient-statistic composer,

$$P(s_{t+1} \mid h_{t+1}) = P(s_{t+1} \mid f(h_t, a_t, o_{t+1})) = P(s_{t+1} \mid h_t, a_t, o_{t+1}) = P(s_{t+1} \mid \tau_{t+1}) \tag{11}$$

which means that h_{t+1} is a sufficient representation of τ_{t+1}. The last equality is true if h_t is a sufficient representation of τ_t.

An example of a sufficient-statistic composer is the encoder of an auto-encoder that learns a one-to-one mapping from an input space to a representation space. In this case, because it is possible to reconstruct (h, a, o) from $f(h, a, o)$, f satisfies the definition of a sufficient-statistic composer.

3 Converting POMDPs to MDPs using History Representation

Policy. To make decisions, the agent maintains a policy $\pi_\theta(a \mid h)$ that maps from a history representation h to a distribution over actions in $\mathcal{A}$.

At test time, each episode proceeds as follows. The agent starts in state s_1 and observes $o_1 \sim O(\cdot \mid s_1, a_0)$, where a_0 is a special `<start>` action. The initial history representation h_1 is computed. At time step t, the agent selects an action $a_t \sim \pi_\theta(\cdot \mid h_t)$, which transitions it to a new state $s_{t+1} \sim T(\cdot \mid s_t, a_t)$ and induces a reward $r(s_t, a_t, s_{t+1})$. It receives a new observation $o_{t+1} \sim O(\cdot \mid s_{t+1}, a_t)$. The next history representation is computed: $h_{t+1} = f(h_t, a_t, o_{t+1})$.

Return. The return of a trajectory is

$$R(\tau) = \sum_{t=1}^{L} r(s_t, a_t, s_{t+1}) \tag{12}$$

if time is finite. In the infinite-time case, the return is

$$R(\tau) = \sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t, s_{t+1}) \tag{13}$$

where γ is a discount factor.

I will construct utility functions for the infinite-case because it is the most widely studied in the literature. This would help readers easily draw the connections with the literature.

Conversion to MDPs. I first define the distribution over the next observations given the current history representation and action:

$$P(o_{t+1} \mid h_t, a_t) = \sum_{s_t, s_{t+1}} P(s_t, s_{t+1}, o_{t+1} \mid h_t, a_t) \tag{14}$$

$$= \sum_{s_t, s_{t+1}} b(s_t \mid h_t) \cdot T(s_{t+1} \mid s_t, a_t) \cdot O(o_{t+1} \mid s_{t+1}, a_t) \tag{15}$$

Given this distribution, I now define the state-transition function and reward function of the new MDP. Because we are going to use history representations as states in the new MDP, these functions take history representations as input:

$$\bar{T}(h' \mid h, a) = \mathbb{E}_{o' \sim P(\cdot \mid h, a)} \left[\mathbb{1} \left\{ f(h, a, o') = h' \right\} \right] \tag{16}$$

$$\bar{r}(h, a, h') = \mathbb{E}_{s,s',o' \sim P(\cdot|h,a)} \left[r(s, a, s') \cdot \mathbb{1} \left\{ f(h, a, o') = h' \right\} \right] \tag{17}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. Notice how the sufficient representation condition allows us to construct a Markovian transition function, where the distribution over the next representation depends only on the current representation.

The tuple $(\mathcal{H}, \mathcal{A}, \bar{T}, \bar{r})$ forms an MDP.

Utility functions. The value function is defined over the space of history representations

$$V^{\pi}(h) = \mathbb{E}_{s \sim b(\cdot|h), \tau \sim P_{\pi}(\cdot|s)} \left[R(\tau) \right] \tag{18}$$

where $P_{\pi}(\tau \mid s)$ is the probability of generating trajectory τ given policy π and start state s.

To see that this is an extended definition of the standard $V(s)$ function defined over states, we view an environment state as a one-hot belief state. In other words, if h is a representation of a single state s, $b(\cdot \mid h) = \delta_s$ is a one-hot distribution that peaks at s. Then,

$$V^{\pi}(h) = \mathbb{E}_{s \sim \delta_s, \tau \sim P_{\pi}(\cdot|s)} \left[R(\tau) \right] = \mathbb{E}_{\tau \sim P_{\pi}(\cdot|s)} \left[R(\tau) \right] \tag{19}$$

which is exactly the definition of the standard $V(s)$ function.

The relationship between $V(s)$ and $V(h)$ is given as follows

$$V^{\pi}(h) = \mathbb{E}_{s \sim b(\cdot|h)} \left[V^{\pi}(s) \right] \tag{20}$$

Using our newly constructed MDP's state-transition and reward functions, I can define the value function recursively:

$$V^{\pi}(h) = \mathbb{E}_{a \sim \pi(\cdot|h), h' \sim \bar{T}(\cdot|h,a)} \left[\bar{r}(h, a, h') + \gamma V^{\pi}(h') \right] \tag{21}$$

The optimal value function is

$$V^{\star}(h) = \max_{\pi} V^{\pi}(h) = \max_{a} \mathbb{E}_{h' \sim \bar{T}(\cdot|h,a)} \left[\bar{r}(h, a, h') + \gamma V^{\star}(h') \right] \tag{22}$$

Similarly, I define the Q function as follows

$$\begin{aligned} Q^{\pi}(h, a) &= \mathbb{E}_{s \sim b(\cdot|h), \tau \sim P_{\pi}(\cdot|s)} \left[R(\tau) \mid a_1 = a \right] && (23) \\ &= \mathbb{E}_{s \sim b(\cdot|h)} \left[Q^{\pi}(s, a) \right] && (24) \end{aligned}$$

where a_1 is the first action of a trajectory. Since we are in an MDP, these identities come for free:

$$\begin{aligned} Q^{\pi}(h, a) &= \mathbb{E}_{h' \sim \bar{T}(\cdot|h,a)} \left[\bar{r}(h, a, h') + \gamma V^{\pi}(h') \right] && (25) \\ V^{\star}(h) &= \max_{a} Q^{\star}(h, a) && (26) \end{aligned}$$

Implementation. While the newly defined MDP is useful for constructing RL algorithms for POMDPs and studying them, it does not tell us how to implement these algorithms efficiently. Specifically, when implementing RL algorithms, we want to be able to draw samples of transition (s_t, a_t, s_{t+1}, r_t). In a regular MDP, this process is straightforwardly done by sampling trajectories: we start from a start state s_1 and at every step draw action $a_t \sim \pi(\cdot \mid s_t)$, where π is some behavior policy, next state $s_{t+1} \sim T(\cdot \mid s_{t+1}, a_t)$, and receive reward $r(s_t, a, s_{t+1})$.

In the POMDP case, the tuples we want to sample are $(h_t, a_t, h_{t+1}, \bar{r}_t)$. It is not straightforward how to sample $h_{t+1} \sim \bar{T}(\cdot \mid h_t, a_t)$ efficiently. To understand why, suppose we apply the sampling process defined by Equation 15

(a) Draw $s_t \sim b(\cdot \mid h_t)$

(b) Draw $s_{t+1} \sim T(\cdot \mid s_t, a_t)$

(c) Draw $o_{t+1} \sim O(\cdot \mid s_{t+1}, a_t)$ (then compute $h_{t+1} = f(h_t, a_t, o_{t+1})$)

(d) Receive reward $r(s_t, a_t, s_{t+1})$, which in expectation is equal to $\bar{r}(h_t, a_t, h_{t+1})$

The main challenge is to sample from the belief state $b(\cdot \mid h)$ efficiently. In principle, we can incrementally compute the belief state along a trajectory using Equation 3. However, when the state space is large, we do not want to construct the full belief state explicitly.

In practice, we sample from the belief state also by sampling trajectories. We start from a start state s_1 and initial representation h_0. At step t, given state s_t, we generate: $o_t \sim O(\cdot \mid s_t, a_{t-1})$, $h_t = f(h_{t-1}, a_{t-1}, o_t)$, $a_t \sim \pi(\cdot \mid h_t)$, $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, and receive $r(s_t, a_t, s_{t+1})$. We then use the tuples (h_t, a_t, h_{t+1}, r_t) for learning. This is a *different* sampling process from the one above. Here, h_t is generated *after* s_t is, whereas in the above sampling process, h_t is generated *before* s_t is. We can understand this difference as sampling from the joint instead of from the conditional

$$b(s \mid h) \propto P(s, h) = P(s)P(h \mid s) \tag{27}$$

Here, the problem is that we cannot directly sample from $b(s \mid h)$ efficiently. Hence we sample from $P(s, h)$, using the process defined by the factorization $P(s)P(h \mid s)$, which implies that we sample s first followed by h.

In the end, the POMDP-to-MDP conversion does not significantly affect the implementation. Like in an algorithm designed for MDPs, we still interact with the environment to generate trajectories and collect transition samples. We also do not need to modify the reward function. The only change is to add a procedure that computes history representations and to use them as input states of the utility functions.

REFERENCES

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15)*, November 2015.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.