POMDP Categorization Based on Agent Memory

Alexander Ivanov

May 15, 2025

1 Introduction

A common assumption for many reinforcement learning problems is that the core environment dynamics are accurately described as a Markov Decision Process (MDP). While this basic assumption has allowed the development of efficient algorithms across a variety of tasks, it is unrealistic and requires that the learning agent has access to the exact world state. In many cases, especially for real world problems, we are interested in learning optimal behavior while not having access to all the information. Partially Observable Markov Decision Processes (POMDP) model the learning agent's incomplete view of the world and so can be applied to a much broader set of problems. In particular, POMDPs can be used to represent tasks were the agent has noisy sensors or where the agent cannot see all parts of its environment, problems that are often seen in robotics where real physical agents need to navigate complex spaces with limited sensors.

POMDPs represent a wide class of problems which makes the impact and value of an efficient learning algorithm quite large. At the same time, because POMDPs represent so many possible tasks, some of those tasks can be incredibly difficult. POMDPs are provably hard to solve and in practice finding solutions can be intractable [6]. A common approach is to define a new subset of POMDPs which admits to practical algorithms because of a shared property or structure [1–3, 10]. Although the resulting POMDP class may not represent all POMDPs, there can still be many practical applications which fall within the POMDP class. Investigating such classes of POMDPs and their relationships is essential for developing new methods and to understanding the space of all POMDPs as a whole.

A specific area that hasn't been as thoroughly investigated with respect to POMDPs is the role of agent memory. Previous work explores similar ideas, such as history features [5, 7, 9] and variable-length history windows [5, 8], but, as we will see there are still many unanswered questions and unexplored ideas. This report will show how agent memory is closely tied with the complexity and structure of POMDPs. Memory types can be used to define various POMDP classes and the relationships between different types of memory suggest the existence of a POMDP class hierarchy. This work aims to investigate the complexity of POMDPs through the lens of agent memory and to uncover classes of POMDPs in terms of the kind of memory used to solve them.

2 Background

2.1 POMDP and Trajectories

POMDPs are defined as a 7-tuple $(S, A, P, R, O, \Phi, \gamma)$ where S is the set of states, A is the set of actions, $P : S \times A \to \Delta S$ is the transition function, $R : S \times A \times S \to \mathbb{R}$ is the reward function, O is the set of observations, $\Phi : S \times A \to O$ is the observation function, and γ is the discount factor. Additionally, there is some initial state distribution ΔS which determines which starting state for each new episode.

When an agent first begins an episode it receives an initial observation as determined by Φ and preforms a series of actions, receiving the corresponding observations and rewards. The sequence of actions and observations is called the agent trajectory and is represented by τ. Rewards are excluded as any POMDP can be trivially modified for the observations to include the reward at the same time step and such modification doesn't change the results or conclusions of this work. We write τ as $(o_0, a_0, o_1, a_1, \ldots, o_n, a_n)$.

We can also consider a partial trajectory τ which represents a sequence of actions and observations which hasn't necessarily reached the end of an episode. For a given partial trajectory where the agent takes action a and receives observation o we can express the resulting partial trajectory as $\tau' = \tau + (a, o)$. Note that if action a resulted in terminating the episode there would be no following observation and the resulting trajectory would terminate in a.

It is important to note that two agents taking the same actions and encountering the same observations but end up in completely distinct underlying states. This work typically considers the existence of a partial trajectory τ which only implies that there is a non-zero probability that a random agent could experience τ.

In same cases, it is useful to consider the underlying states that an agent traverses in addition to the actions and observations. I represent this as a true trajectory and write it as $\bar{\tau} = (s_0, o_0, a_0, s_1, o_1, a_1, \ldots)$. It can also be useful to consider states and actions alone which I also refer to as a true trajectory and write similarly as $\bar{\tau} = (s_0, a_0, s_1, a_1, \ldots)$. I typically only consider trajectories and partial trajectories and will make clear when I am using true trajectories or true partial trajectories and if they include observations.

2.2 Reinforcement Learning Agent With Memory

An agent interacting with an environment, specifically a POMDP, can be represented by a policy $\pi : O \to \Delta A$ which for a given observation o samples an action $a \sim \pi(o)$. We also write $\mathbb{P}(a|o)$ to represent the probability of a particular action under the policy. A call a policy deterministic if for all $o \in O$ there exists $a \in A$ such that $\mathbb{P}(a|o) = 1$. Otherwise the policy is stochastic.

In this work I consider a particular model of a learning agents memory which is sufficiently flexible for our needs; memory as a Finite State Machine (FSM). Under this model the agent has a memory state $m \in M$ and updates this

memory at every time step according to a memory function $\mu : M \times A \times O \to \Delta M$ which takes the current memory state, the previous observation, and the previous action to produce the memory state for the current time step. Like the policy, the memory function can be be stochastic or deterministic. The size of a memory function μ refers to the number of possible memory states $|M|$.

3 Trajectory Sets

When considering how an agent may use its memory to solve a POMDP, the requirements of the memory can be viewed in two ways. One common view is to consider the history of the agent thus far, the partial trajectory τ. If the agent remembers all of its past actions and observations it has all the information it could possibly have and so can pick next action optimally [1]. Another perspective is to consider the result of the agents future actions. If the agent knows the exact distribution of outcomes for any possible choice of actions it also has all the information it needs to pick the next action optimally. I begin by exploring this second case, looking at the future results of agent actions, and eventually come to define trajectory sets.

3.1 Construction

Given a trajectory τ from time step 0 to t with observation o_i at time step i, I define the future of this trajectory as the probability distribution over observations for all finite sequences of actions taken from the end of τ. More precisely, the future of τ is the function $f_\tau : A^n \to \Delta(O)$ such that $f_\tau(a_t, \ldots, a_{t+n}) = \Delta O_{t+n+1}$ a distribution over the possible observations at time step $t + n + 1$. Each observation in O_{t+n+1}, o_{t+n+1} has probability $P(o_{t+n+1}|\tau, a_t, \ldots, a_{t+n})$ given the finite action sequences $a_t, \ldots, a_{t+n}$. If a given action sequence is invalid then the function is said to be undefined. An action sequence can be invalid if the episode terminates earlier in the action sequence or if a particular action is invalid at a particular time step.

We can get an exact expression for the future by relying on the underlying state. Given a true trajectory $\bar{\tau}$ over states and actions, let the set of possible observed trajectories be $O(\bar{\tau})$. Similarly, for a trajectory τ the set of possible true trajectories that could have produced it is $O^{-1}(\tau)$. This lets us write the future of a trajectory τ as:

$$f_\tau(a_0, \ldots, a_n) = \mathop{\mathbb{E}}_{\bar{\tau} \in O^{-1}(\tau)} \left[\mathop{\mathbb{E}}_{\tau' \in O(\bar{\tau} + (a_0, \ldots, a_n))} \left[p(o_n|\tau') \right] \right] \tag{1}$$

Two futures f_1 and f_2 are equal iff $f_1(a_0, \ldots, a_{n-1}) = f_2(a_0, \ldots, a_{n-1})$ for all finite action sequences $a_0, \ldots, a_{n-1} \in A^n$ and both f_1 and f_2 are defined for the same set of finite action sequences. This allows us to define a relation over

[1] The agent does not have access to the underlying state so the only information that is available is the sequence of actions and observations.

all partial trajectories in terms of when the futures of two trajectories are equal. This gives us the definition of trajectory sets.

I define the trajectory set T_τ induced by a trajectory τ be the set of trajectories for which the future is the same:

$$T_\tau = \{\tau' | f_{\tau'} = f_\tau\}. \tag{2}$$

Trajectory sets are a partition of the space of all trajectories. If two trajectory sets T_1 and T_2 share a trajectory τ then all trajectories in both must have future f_τ and so $T_1 = T_2$. Because of this, I can define the future of a trajectory set T as the future of any of its member trajectories, $f_T = f_\tau \ \forall \tau \in T$.

3.2 Markov Property

An interesting property of trajectory sets is that they preserve the Markov property under a natural extension of the POMDP transitions.

Consider an state machine with the states $\mathcal{T} = \{T_\tau | \forall \tau\}$ and actions A. The transition function $\mathbb{P}(T_{\tau'} | T_\tau, a)$ would follow from $\tau' = \tau + (a, o)$ where the probability of $o \in O$ is determined by the POMDP transition probabilities conditioned on the trajectory seen so far: $\mathbb{P}(o|\tau, a)$. As stated, it is not clear that the Markov Property should hold as for each transition there are many choice of $\tau \in T_\tau$.

Given a trajectory set T and an action a, we want to show that all trajectories $\tau \in T$ lead to a single trajectory set T' upon taking action a and receiving observation o. Consider trajectories $\tau \in T$ and $t' \in T'$ with $\tau' = \tau + (a, o)$. We can write $f_{\tau'}$ exactly as $f_{\tau'}(a_1, \ldots, a_n) = f_T(a, a_1, \ldots, a_n | o)$ and we notice that this holds for all $\tau \in T$ because they all by definition have the same future f_T. This tells us that the transition function is well defined.

Because we have that the transitions are only dependent on the most recent trajectory set, $p(T_{n+1} | T_0, T_1, \ldots, T_n, a) = p(T_{n+1} | T_n, a)$, we have that the Markov property is preserved.

3.3 Trajectory Set MDP

There are two possible ways to construct an MDP with trajectory sets as states. We can exclude observations from the action set as we see in section 3.2 which results in a stochastic MDP. The transitions are stochastic because the probability of observation o for action a in a partial trajectory τ depends on the distribution of true trajectories $\bar{\tau}$ which could generate τ and the possible observations that could be produces from taking action a in those true trajectories.

A more interesting MDP construction incorporates the observation into the available actions. In this case the states of the POMDP are still trajectory sets but the actions are action-observation tuples (a, o) resulting in deterministic transitions. Starting in trajectory set T_τ and taking action (a, o) results in exactly trajectory set $T_{\tau+(a,o)}$. An important caveat here is that not all actions are available in all trajectory states. For a trajectory set T_τ the only available

actions (a, o) are those where the probability of the observation o given a and some partial trajectory $\tau \in T_\tau$ is non-zero. This draw back makes this MDP hard to analyze.

Reward for both cases is hard to define and can be expressed in terms of posterior distributions over the underlying state. We could consider a possible reward function such as:

$$R(T_\tau, a, T_{\tau'}) = \mathop{\mathbb{E}}_{\bar{\tau} \in O^{-1}(\tau)} \mathop{\mathbb{E}}_{\bar{\tau}' \in O^{-1}(\tau')} R(s, a, s')$$

with s and s' being the final states of $\bar{\tau}$ and $\bar{\tau}'$ respectively. Similarly, we would have the following expression when actions are an $A \times O$ tuple.

$$R(T_\tau, (a, o), T_{\tau+(a,o)}) = \mathop{\mathbb{E}}_{\bar{\tau} \in O^{-1}(\tau)} \mathop{\mathbb{E}}_{\bar{\tau}' \in O^{-1}(\tau+(a,o))} R(s, a, s')$$

In each case there is a tricky dependence on the underlying state which makes this reward definition impractical. Additionally, it is an average over possible rewards and so wouldn't correspond to the actual reward an agent might see when traversing an environment and consider which trajectory set it moved from and to.

3.4 Information Constant Trajectory Sets

while constructing a well defined and cohesive MDP in terms of trajectory sets is difficult, we still see that trajectory sets can provide useful insight into how the information an agent holds about it environment. In particular, we see that trajectory sets in some way represent both information about the location of the agent in an environment and also what the agent could possibly know about its environment based on its history.

We define $\tau \to_n \tau'$ as a predicate indicating the existence of a sequence of actions of length n, $a_0, \ldots, a_{n-1}$, such that taking the actions from τ can result in trajectory τ' with non-zero probability. If $\tau \to_n \tau'$, τ' is n steps in the future from τ.

We can define an analogous relation on trajectory sets:

$$T_1 \to_n T_2 := \forall\, \tau \in T_1\ \exists\, \tau' \in T_2\ \textit{s.t.}\ \tau \to_n \tau'. \tag{3}$$

We write $T_1 \to T_2$ as shorthand for $T_1 \to_1 T_2$.

We say that two trajectory sets T_1 and T_n are information constant, written as $T_1 \sim T_n$, if $T_1 \to T_n$ and $T_n \to T_1$ or there exists a sequence $T_1, T_2, \ldots, T_{n-1}, T_n$ such that $T_i \sim T_{i+1}$ for all $i \in [1, n-1]$.

Intuitively, a trajectory set T_1 represents the future of a particular trajectory and writing $T_1 \to T_2$ for $T_1 \neq T_2$ implies some change in the probability distributions of future observations for the agent. Consider the example in figure 3.

Trajectories that end in different corridor states would have different trajectory sets because the colored state is a different number of actions away. This

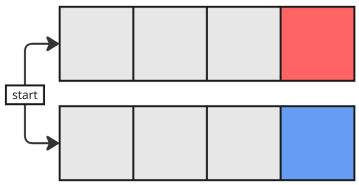

Figure 1: Hallway POMDP with three hallway states and a single end state. The agent starts in either the top or bottom hallways. The corridor states all have the same observation, o_{grey}, and the end state gives either observation o_{red} or o_{blue} depending on the hallway.

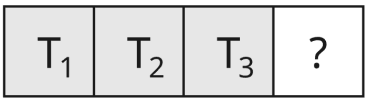

Figure 2: Hallway POMDP with three labeled trajectory sets before the color of the end state is discovered.

means we can write $T_1 \to T_2$ and $T_2 \to T_3$ because the color of the end state is unknown. We also have that $T_3 \to T_2$ and $T_2 \to T_1$ so $T_1 \sim T_2 \sim T_3$. Intuitively, moving up and down the corridor changes the future because the end state is a different number of steps away but it doesn't resolve the color of the end state, making these trajectory sets information constant.

disc

This example also shows what it means for trajectory sets to not be information constant. By walking to the end of the corridor the color is observed and the agent now knows if it is in the top or bottom hallway. This new information is reflected in the futures of trajectories that saw the top corridor versus the bottom. While both $T_3 \to T_4$ and $T_3 \to T_8$ are true, we don't have the reverse direction because that would correspond to forgetting the color of the end state. We do however have that $T_7 \sim T_6 \sim T_5 \sim T_4$ and $T_{11} \sim T_{10} \sim T_9 \sim T_8$.

Figure 3: Hallway POMDP trajectory sets after the color of the end state is discovered.

3.5 Trajectory Sets and Memory

Trajectory sets have some useful properties with respect to representing the location of an agent and its information about the environment. It turns out that trajectory sets actually translate naturally into a deterministic memory function which can achieve optimal behavior along with some policy.

Notice that for a given POMDP, a learning agent that represents each trajectory set with a memory state would be just as good as a learning agent that remembers the entire trajectory. If we consider two partial trajectories τ_1 and τ_2 which fall into the same trajectory set T we realize that the optimal action must be the same for both because by definition their futures are indistinguishable. There may exist an optimal policy π^* and memory function μ^* for which τ_1 and τ_2 are followed by distinct actions but taking the same action must also be optimal.

Once we have memory states, M, for each trajectory set we can also define the memory function transitions in terms of the deterministic version of the trajectory set MDP presented in section 3.3. At each time step the agent takes an action a and observes observation o. To update their memory state m consider the corresponding trajectory set T, take any trajectory $\tau \in T$ then find the trajectory set T' for $\tau + (a, o)$ and the corresponding memory state m'. This gives an exact definition for the deterministic memory function $\mu : M \times A \times O$.

One important note is that the number of trajectory sets for the POMDP is not necessarily finite and so this memory function may require an infinite number of memory states. Nonetheless, the fact that such a memory function exists and can be deterministic, is an interesting result. This motivates us to consider if there are possible steps that can be taken to reduce the necessary number of memory states. Specifically, we want the smallest number of memory states for which there still exist a memory function and policy pair, (μ, π), that match the optimal policy π^* with a memory function μ^* that remembers the entire trajectory thus far τ.

In general, to reduce the necessary number of memory states $|M|$ for a memory function μ, without changing its behavior, we can carefully inspect the policy and memory function. If for two memory states $m_1, m_2 \in M$ and all observations $o \in O$ we that $\pi(m_1, o) = \pi(m_2, o)$ then the distinction between m_1 and m_2 is not necessary from the perspective of the policy. Similarly, if for two memory states $m_1, m_2 \in M$, all observations $o \in O$, and all actions $a \in A$ we that $\mu(m_1, a, o) = \mu(m_2, a, o)$ then the distinction between m_1 and m_2 is not necessary from the perspective of the memory function. If distinguishing m_1 and m_2 is not necessary for both the policy and the memory function then the two memory states can be treated as the same reducing the number of required memory states. We can additionally restrict our view to only the trajectories that the agent has a non-zero probability of taking under π and μ. This allows us to consider a subset of all actions and observations when determining if two memory states need to be distinguished which can further reduce the required memory size.

While such reductions may reduce the memory size it is not clear if they will

result in the smallest possible memory and in fact, we know that this is not the case by counter example. Note that before any reductions each memory state would correspond to a trajectory set and any further reductions only serve to merge memory states. To show this is not optimal we would need the optimal memory for some POMDP to "split" a trajectory set, taking different actions in different partial trajectories.

Before presenting the counter example, I want to point out how strange this notion is intuitively. The claim is that there exists a POMDP where the agent can walk down two possible paths τ_1 and τ_2 which happen to fall into a singe trajectory set T. By definition, there no sequence of actions the agent could possibly take to distinguish τ_1 from τ_2 and yet, the agent must take different actions in the two partial trajectories. This is incredibly strange and highlights that while there is no reason to distinguish trajectories within a trajectory set with respect to the policy the need does arise when determining the smallest optimal memory function.

Consider a POMDP where an agent receives a sequence of 5 observations in $\{o_0, o_1\}$ and then the o_{count} observation. The agent must count the number of o_1 observations and summarize the count with its two actions $A = \{a_{\text{low}}, a_{\text{high}}\}$. The agent receives positive reward after the last observation if it correctly takes action a_{low} for a count less than 3 and a_{high} for a count greater than or equal to 3. Note that this environment results in at least 5 trajectory sets because the distance from the end of the episode is distinguishable and so partial trajectories at each time step have different futures. We also have a trivial and optimal counting memory function with 4 states where each state represents the number of o_1 seen so far.

Now lets consider the two partial trajectories $\tau_1 = o_0, o_0, o_0, o_0$ and $\tau_2 = o_0, o_0, o_0, o_1$. In both cases, regardless of the third observation, these trajectories will have a count less than 3, will have the same optimal actions, and fall into the same trajectory set. Because they are in the same trajectory set, any memory function that is constructed in terms of trajectory sets would represent both with a single memory state. On the other hand the trivial counting memory function does distinguish these two trajectories. These two trajectories have exactly the desired property because our trivial counting memory distinguishes them while trajectory sets suggest no reason to do so. For this POMDP, no aggregation of trajectory sets into memory states can achieve a memory of the same size as the trivial counter while preserving the optimal policy.

3.6 Practical Considerations

In this section I have investigated various theoretical properties of trajectory sets and how they relate to agent memory functions. Although these are interesting results, for example the existence of deterministic memory functions, there are several drawbacks that make it difficult to consider trajectory sets for practical applications.

Firstly, trajectory sets are defined in terms of the future function. This by itself is a complex object, representing distributions for all possible action

sequences, making it difficult to handle in theory and exceptionally hard to implement in practice. Representing the future of a trajectory in computer memory for a toy POMDP seems daunting let alone a POMDP that has infinite actions sequences. Additionally, some aspects of trajectory sets like the trajectory set MDP still rely in part on the underlying state representation. This further increases the challenge of using trajectory sets in practice because the underlying state may not be available.

There is some hope for trajectory set usefulness in considering the process of learning a new memory function. This work does not investigate this idea in depth but it may be possible to learn a memory function iteratively by relying on the Markov assumption of trajectory sets and the deterministic transitions of the trajectory set MDP. If memory states are assumed to correspond to trajectory sets, one could imagine checking candidate memory functions and updating them iteratively if they violate the deterministic transition property. Investigate algorithms for memory learning based on trajectory sets and further analyzing the theoretical properties of trajectory sets are open areas for future work.

4 Finite Memory Functions

When considering the requirements of agent memory for a POMDP trajectory sets provide a kind of bound on what information a memory function could choose to represent. Unfortunately this model has little promise of translating into practice, in particular because of a dependence on the underlying state and the difficult problem of translating trajectory sets to a well defined memory function. Both of these problems can be addressed by directly considering a finite memory function and determining if it is sufficient for maximizing return or any other metric of choice.

A few metrics that are of particular interest for memory functions come from the state abstraction hierarchy [4]. Memory, and an agents memory function, can be thought of as an abstraction over a particular kind of MDP; the trajectory MDP. This MDP represents the best possible memory an agent could have, remembering everything, and allows us to consider memory functions that preserve particular properties of this MDP.

Definition 4.1 (Trajectory MDP)**.** Given a POMDP $(S, A, P, \gamma, O, \Phi, R)$ with initial state distribution s_0, we define the trajectory MDP to be (T, A, P', R', γ), where $T \coloneqq \{\tau \in (O \times A)^* \times O\}$ is the space of observation-action partial trajectories, $P : \tau \times a_t \mapsto \tau \oplus a_t \oplus o_{t+1}$ with $o_{t+1} \sim \Pr(\cdot|\tau, a_t)$ and $\oplus$ denoting concatenation, and $R'(\tau = (o_0, a_0, \ldots, o_t), a_t) \coloneqq \mathbb{E}_{s_t|\tau}[R(s_t, a_t)]$.

This decision process is Markov by definition as a trajectory τ_t up to time t being a prefix of τ_{t+j} implies that $\Pr(\tau_{t+k}|\tau_t, \tau_{t-1}, \ldots) = \Pr(\tau_{t+k}|\tau_t)$.

In this section I will consider memory functions in the context of three specific metrics derived from the state abstraction hierarchy; expected return

($\pi^* - preservingabstraction$), Q^* error, and Model error. The precise definitions of these metrics are presented in table 1. An interesting consequence of considering these three metrics is that we can show a hierarchical relationship between memory functions that preserve each of the quantities. In particular, we have that a model preserving memory function is also Q^* preserving and a Q^* preserving memory function is in turn π^* preserving.

Type	**State Abstraction**
Model	$\exists f_P : \phi(T) \times A \to \Delta O. \|\|[f_P]_\phi - P_o\|\|_1 < \epsilon_P$ $\exists f_R : \phi(T) \times A \to \mathbb{R}. \|\|[f_R]_\phi - R\|\|_\infty < \epsilon_R$
Q^*	$\exists f : \phi(T) \times A \to \mathbb{R}. \|\|[f]_\phi - Q^*_M\|\|_\infty \leq \epsilon_{Q^*}$
π^*	$\exists \pi : \phi(T) \to \Delta A. \|\|V_M^{[\pi]_\phi} - V^*_M\|\|_\infty \leq \epsilon_{\pi^*}$

Table 1: Approximate State Abstractions for memory functions. Here, $\phi : T \to \Delta M \times \Delta O$, and $P_o(o_{t+1}) = \Pr(o_{t+1}|\tau_t, a_t)$ and R are defined as in Definition 4.1.

In addition to these three metrics I will also consider a few types of memory function. The first distinction is if the memory function itself is stochastic, $\mu : M \times A \times O \to \Delta M$, or deterministic, $\mu : M \times A \times O \to M$. I will also consider the number of memory states a memory function uses (size). For simplicity I will consider either memory functions with 2 states, the smallest possible memory, or k states for arbitrary k.

For a given type of memory function and a given metric we can classify the quality of a memory function into three broad categories.

- Useless - the memory function is no better than no memory with respect to the metric
- Improving - the memory function is better than Useless with respect to the metric
- Optimal - the memory function achieves the maximum possible value of the metric

With these definitions we can begin asking questions about the existence of different kinds of memory functions such as 2-state deterministic Model-improving or k-state stochastic Q^*-optimal.

4.1 Existence of Memory Functions

In general it is hard to say if a given type of memory function must exist for a given POMDP, in fact, it typically does not. We instead restrict the set of POMDPs to those where one kind of memory function exists and we can then consider if another kind must also exist. Each result of this form can be thought of as one type of memory implying the existence of another for the space of POMDPs. These kinds implications allow the construction of a POMDP

class hierarchy where each class of POMDP corresponds to a particular type of memory function. A structured hierarchy of this kind could be incredibly useful for future research as it would carve up the large and intractable space of POMDPs into more manageable chunks and the hierarchical nature may aid in generalizing results to specific sub categories of POMDP more easily. The following examples and proofs are motivated by the construction of such a hierarchy.

4.1.1 Detecting Stochastic Memory - Expected Return

For this first counter example we consider a POMDP which is adversarially designed to reward agents with stochastic memory. By relying on the probabilistic nature of stochastic memory this POMDP statistically analyses the agents actions and can detect when memory transitions occur deterministically. The statistical test built into the POMDP can only detect deterministic memory in the limit and so it relies on unbounded reward to outpace the exponential discounting which normally pushes return to zero for the ends of long trajectories. This counter example shows that while stochastic memory can help improve expected return, deterministic memory may be of no use.

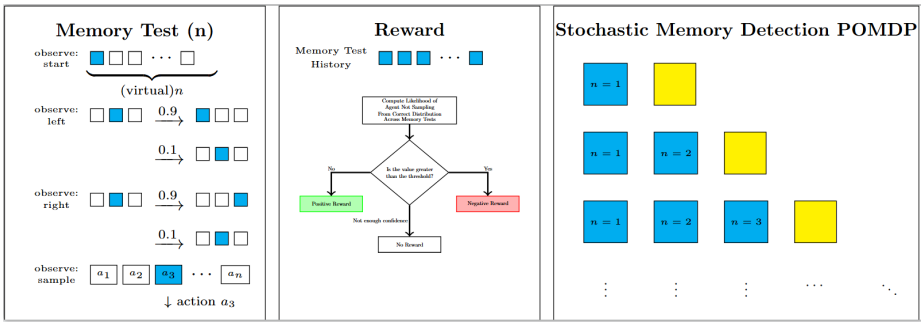

Figure 4: Diagram of counterexample POMDP. Detection mechanism (left). Reward mechanism (center). Combination of detection and reward (right).

The following counterexample proves the following claims:

- The existence of 2-stochastic expected return optimal memory doesn't imply the existence of k-deterministic expected return optimal memory if rewards are unbounded.
- The existence of 2-stochastic expected return improving memory doesn't imply the existence of k-deterministic expected return improving memory if rewards are unbounded.
- The existence of 2-stochastic or k-stochastic expected return optimal memory doesn't imply the existence of k-deterministic expected return optimal memory if rewards are unbounded.

- The existence of 2-stochastic or k-stochastic expected return improving memory doesn't imply the existence of k-deterministic expected return improving memory if rewards are unbounded.

To show both the optimal conditions and the improvable conditions it is sufficient to show that for a given POMDP where there exists a k-stochastic memory function which is expected return optimal there doesn't necessarily exist a k-deterministic memory function which is improvable. This is derived from the fact that an optimal memory function is improving, along with its contrapositive: if there doesn't exist an improving memory function there cannot exist an optimal one. For this purpose, we construct the following counter example and depict its structure in Figure 4.

Consider an environment where the agent's task is to simulate trajectories in a stochastic virtual environment and sample from the state distribution after some number of steps. The agent's actual environment is constructed adversarially such that the agent is rewarded if it produces different samples for the same simulated trajectory over multiple trials. The environment is composed of a detection mechanism and a rewarding mechanism. The detection mechanism detects whether an agent has deterministic or stochastic memory irrespective of the stochasticity of the policy. The rewarding mechanism produces different reward based on if the agent memory is deterministic or stochastic and induces non-improving reward for deterministic memory. Finally, there is also an opt-out action that can be taken at the first time step giving the agent 0 reward.

The detection mechanism is composed of a series of tests where the agent is asked to simulate a particular stochastic environment. The virtual environment is a board with $n = 10$ spaces in a line and a token in the first space. The value of n corresponds to the number of memory states of the stochastic memory function and we choose it to be 10 for the purposes of this example. For an agent with 2 stochastic memory states we would similarly have $n = 2$. Each test starts with o_{reset} which indicates that the token should be virtually placed on the first space. Then, an arbitrary long sequence of observations from the set $\{o_{\text{left}}, o_{\text{right}}\}$ is provided to the agent. When receiving o_{left} or o_{right} the agent is expected to simulate the token moving left or right respectively with probability 0.9 or otherwise staying in place (the choice of probability here is arbitrary as long as it isn't uniform). Finally, the agent gets the observation o_{sample} for which the agent is expected to take an action from $a_1, \ldots, a_{10}$ corresponding to where the simulated token ended up. For all other observations, the agent is expected to provide the action a_0.

After a single test the likelihood that the sampled action was in fact from the expected virtual distribution is computed and by repeating the test the statistical confidence can be increased. To run infinitely many tests infinitely many times a list of current tests is constructed and run sequentially. Upon completion a new longer test is added and the full list of tests is repeated. This ensures that in the limit as a finite time step t goes to infinity, infinitely many tests, are run infinitely many times, and the length of the tests also approaches infinity.

A single test cannot be executed perfectly by an agent with deterministic memory while it is trivially handled by a stochastic memory agent with 10 memory states. For any choice of finite deterministic memory size there will eventually be a test that requires remembering more possible distributions than there are memory states. In this case, the best that the agent would be able to do is sample from a distribution that is ϵ close to the true distribution. As that particular test is repeated infinitely many times, the discrepancy between the agents sampling distribution and the true distribution will always become statistically significant and detectable.

For the rewarding mechanism of the environment, we simply give the agent reward depending on if it is believed to have stochastic or deterministic memory. A reward that is exponential in the time step, $|R(t)| = O(1/\gamma^t)$, would be sufficient to overshadow the discount factor γ. For this particular counter example, we will provide a positive reward for agents believed to have stochastic memory and a negative reward for agents believed to have deterministic memory (according to the detection mechanism). Positive and negative rewards are equal in magnitude. If the current likelihood estimate doesn't have sufficient confidence a reward of 0 is given. In the limit, the probability that the detecting mechanism is wrong about the nature of the agents memory goes to zero and so in the limit, all agents with receive positive/negative reward if they have stochastic/deterministic memory respectively.

The opt-out action is available for the agent at the first time step and if taken gives the agent 0 reward and ends the episode. This reward is arbitrary as long as it is better than the reward achieved by an agent with deterministic memory. This ensures that the best option for a deterministic agent is to opt out and so achieve the same performance as no memory.

We now combine the detecting mechanism with the rewarding mechanism. Importantly, the detecting piece is never certain that a given agent has deterministic or stochastic stochastic memory for any finite time step t. This means we cannot switch to the rewarding piece indefinitely. Instead, after each test in the detecting piece we allow the rewarding piece to take over for a single time step to provide reward based on the current likelihood of the agent having deterministic memory. Because of randomness an agent with stochastic memory may be believed to have deterministic memory but in the limit this will resolve and the expected return will be positive. A deterministic memory agent with finitely many states will be detected after finitely many time steps and so will have a negative expected return even if it receives zero or positive reward for some finite number of time steps initially. Because of this, any non-stochastic agent or agent with insufficient memory will take the opt-out action when maximizing expected return and so achieve the same reward as a no memory function behavior.

Note that for this counter example we require non-finite trajectories and we only detect deterministic memory in the limit as the time step goes to infinity. If trajectories are finite, then proof 4.1 gives a deterministic memory that is better or equal to any given stochastic memory.

4.1.2 Imitating Stochastic Memory - Expected Return

The previous counter example relies on unbounded rewards in order to separate deterministic memory from stochastic memory. It turns out however that if rewards are bounded, k deterministic memory can achieve arbitrarily close performance to k stochastic memory by imitating the stochastic system. Moreover, if trajectory length is finite this proof shows that k deterministic memory is just as good if not better than stochastic memory for maximizing expected return.

This result actually states something more general about finite automata and not just memory functions and so I present the lemma and proof in those terms. For our purposes a deterministic finite automata (DFA) and stochastic finite automata (SFA) are equivalent to deterministic and stochastic memory functions respectively.

Lemma 4.1. *Let μ_k^* be a k-state stochastic finite automata that will serve as a memory function in a POMDP. For any POMDP with bounded reward and for all ϵ, there exists a k'-DFA which achieves an expected return that is only ϵ less than μ_k^*. Furthermore, it is sufficient to choose $k' \geq k \ln(\epsilon(1-\gamma)/R_{ax})/\ln(\gamma)$ where R_{max} is the bound on reward and γ is the discount factor.*

Proof. Let μ_k^* be the given k-SFA memory function with the corresponding policy π^*. Let $\hat{\mu}_{k'}$ be the k'-DFA memory function with corresponding policy $\hat{\pi}$.

For a given POMDP, let τ_t be a trajectory in the environment of states, observations, memory states, actions and rewards up to time step t where memory state m_t is being chosen. Let the observation, memory states, and rewards for a time step t be o_t, m_t, and r_t, respectively.

For a given τ_t, we define $G_{\pi,\mu}(\tau_t)$ as the expected sum of discounted rewards for trajectories that start with τ_t and then proceed according to the policy π and memory function μ.

$$G_{\pi,\mu}(\tau_t) = \mathop{\mathbb{E}}_{\tau|\tau_t}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r_i\right]$$

We then define $G_{\pi,\mu}(m_t, \tau_t)$ as the expected sum of discounted rewards for trajectories that start with τ_t, transition to memory state m_t at time step t, and then proceed according to the policy π and memory function μ.

$$G_{\pi,\mu}(m_t, \tau_t) = \sum_{\tau_{t+1}} \Pr(\tau_{t+1}|\tau_t, m_t) G_{\pi,\mu}(\tau_{t+1})$$

where $\Pr(\tau_{t+1}|\tau_t, m_t)$ is the probability of a trajectory of length $t+1$ given that it starts with trajectory τ_t of length t and that the memory state at time step t is m_t given the policy π.

Let $P(m'|m, a, o)$ be the probability distribution for the transitions of the memory function μ. This gives $P^*(m'|m, a, o)$, the probability distribution of the stochastic memory function μ_k^*, and $\hat{P}(m'|m, a, o)$, the probability distribution of the deterministic memory function $\hat{\mu}_{k'}$.

For any time step t we can write the expected return of the stochastic policy as:

$$G_{\pi^*,\mu_k^*} = \mathop{\mathbb{E}}_{\tau_t}\left[\sum_{m_t \in M} P^*(m_t|o_t, a_{t-1}, m_{t-1}) G_{\pi^*,\mu_k^*}(m_t, \tau_t)\right]$$

Because M is finite, there must exist a $\hat{m}_t$ such that for all possible $m_t \in M$

$$G_{\pi^*,\mu_k^*}(\hat{m}_t, \tau_t) \geq G_{\pi^*,\mu_k^*}(m_t, \tau_t)$$

We then let $\hat{P}(\hat{m}_t|o_t, a_{t-1}, m_{t-1}) = 1$ and have $\hat{p}$ be 0 for all other m_t. This guarantees that

$$\mathop{\mathbb{E}}_{\tau_t}\left[\sum_{m_t \in M} P^*(m_t|o_t, a_{t-1}, m_{t-1}) G_{\pi^*,\mu_k^*}(m_t, \tau_t)\right] \leq$$
$$\mathop{\mathbb{E}}_{\tau_t}\left[\sum_{m_t \in M} \hat{P}(m_t|o_t, a_{t-1}, m_{t-1}) G_{\pi^*,\mu_k^*}(m_t, \tau_t)\right]$$

Note that such assignment of $\hat{P}$ is equivalent to a deterministic memory function. So we have that for a particular time step t the memory state can be chosen in a deterministic way to achieve the same or better expected return when compared to choosing the memory state according to μ_k^*

The same argument can be made inductively, conditioning on a finite initial trajectory τ_{start}. We can consider longer and longer starting trajectories and in each case we can deterministically assign $\hat{P}$ to achieve the same or better expected return when compared to μ_k^*.

$$\begin{aligned}\mathop{\mathbb{E}}_{\tau_t|\tau_{start}}\left[\sum_{m_t \in M} P^*(m_t|o_t, m_{t-1}) G_{\pi^*,\mu_k^*}(m_t, \tau_t)\right] \leq \\ \mathop{\mathbb{E}}_{\tau_t|\tau_{start}}\left[\sum_{m_t \in M} \hat{P}(m_t|o_t, m_{t-1}) G_{\pi^*,\mu_k^*}(m_t, \tau_t)\right]\end{aligned} \tag{4}$$

where $E_{\tau_t|\tau_{start}}$ is the expectation over trajectories τ_t that start with τ_{start}. Importantly, this holds only for finite trajectories τ_{start}. Consider picking the memory states deterministically as described for trajectories τ_{start} of increasing length. Equation 4 will continue to hold and at some finite point the $G_{\pi^*,\mu_k^*}(m_t, \tau_t)$ term will become epsilon small due to the bounded reward. This means that a deterministic memory can achieve the same or better expected return compared to the stochastic memory function for a finite number of time steps t and after is ϵ close. To achieve this however, we need to distinguish identical observation, action, memory state pairs that might occur when considering trajectories of different lengths. To remedy possible conflicts that would prevent always selecting the optimal memory transitions, we can augment the memory with the current time step t.

We construct $\hat{\mu}_{k'}$ by making t copies of each memory state in μ^*_k, one for each of the first t time steps. So for a given memory state m from μ^*_k we now have m_t for each time step t. The policy $\hat{\pi}$ can be defined to return the same action as π^* for each of the t duplicates of a given memory state, ignoring the time step. We then construct $\hat{\mu}$ as described above by taking the best choice of memory state transition at each time step ensuring that $G_{\pi^*,\mu^*_k} \leq G_{\hat{\pi},\hat{\mu}_{k'}}$.

This gives us a k'-deterministic memory function with $k' = k * t$. To guarantee $G_{\pi^*,\mu^*_k} - G_{\hat{\pi},\hat{\mu}_{k'}} < \epsilon$ for a given ϵ we can consider the worst case which would be a difference of $R_{\max}$ right after the first t time steps. This gives the expression $R_{\max}\gamma^t(1 + \gamma + \gamma^2 + \dots) \leq \epsilon$ which means it is sufficient to take t greater than $\ln(\epsilon(1-\gamma)/R_{\max})/\ln(\gamma)$. This works because once the deterministic memory function matches the performance of the stochastic memory function for all trajectories of a sufficiently large finite length, all further rewards are negligibly small due to the discount factor γ.

□

4.1.3 Model and Q^* Error Bounds

Before continuing to further results for Model and Q^* error I prove a simple intermediary result. Relying on the fact that both Q^* abstractions and Model abstractions share a requirement for the memory function to represent reward/return I produce the following result:

Lemma 4.2. *If there exists a terminal trajectory, τ, such that $|[f]_\phi(\tau) - R(\tau)| \geq \epsilon$ for all $f : \phi(T) \times A \to \mathbb{R}$, then:*

1. $\epsilon_{Q^*} \geq \epsilon$
   Because τ is a terminal trajectory $Q^*_M(\tau) = R(\tau)$ and by the definition of infinity norm $||\cdot||_\infty$, we must have that ϵ_{Q^*} is at least ϵ

2. $\epsilon_R \geq \epsilon$
   By the definition of infinity norm $||\cdot||_\infty$, ϵ_R must at least be ϵ

This result lets us simplify the search for counter examples that violate a particular Q^* error or Model error condition. By finding a single terminal trajectory that satisfies the condition we can immediately bound both ϵ_{Q^*} and ϵ_R. Constructing a POMDP to contain such a trajectory is typically an easier task and we see a couple such examples in the following sections.

4.1.4 Deterministic Memory Compounding Error - Model and Q^*

This counter example presents a POMDP for which 2 stochastic memory states are sufficient to be both model and Q^* optimal while no amount of deterministic memory can help at all. In particular, this POMDP highlights how stochastic memory can be highly expressive, not because of its state at any particular time step, but because of the real-valued probabilities defining the memory transitions.

The following counter example is for the following results:

- The existence of 2-stochastic Q* improving memory doesn't imply the existence of k-deterministic Q* improving memory.
- The existence of 2-stochastic Model improving memory doesn't imply the existence of k-deterministic Model improving memory.
- The existence of 2-stochastic Q* optimal memory doesn't imply the existence of k-deterministic Q* optimal memory.
- The existence of 2-stochastic Model optimal memory doesn't imply the existence of k-deterministic Model optimal memory.

First we define a virtual MDP with two states s_1 and s_2 and a parameterized set of actions $A = \{a_x | x \in [-1, 1]\}$. Actions a_x with $x >= 0$ result in the the following two transitions $P(s_2|s_1, a_x) = x$, $P(s_1|s_1, a_x) = 1 - x$, and $P(s_2|s_2, a_x) = 1$. Actions a_x with $x < 0$ result in the the following two transitions $P(s_1|s_2, a_x) = x$, $P(s_2|s_2, a_x) = 1 - x$, and $P(s_1|s_1, a_x) = 1$. Actions are selected uniformly at random at each time step.

We now wrap this MDP with a POMDP to produce the desired counter example. The POMDP tracks the running probability of state s_1 and at each time step communicates the action taken in the MDP, a_x, to the agent as observation o_x. At each time step the POMDP has a .1 probability of terminating and presenting the agent with the o_{end} observation. For this observation the reward is equal to the probability of the MDP being in state s_1. The reward for all other observations is 0. The trajectory terminates after the o_{end} observation. The action space for the agent is $A = \{a\}$, a single action for all time steps.

There exists a 2-stochastic optimal memory which is sufficient to predict the reward at each time step. Specifically, we take the memory function which for observations transitions its memory states m_1 and m_2 in the same way as the virtual MDP transitions its states s_1 and s_2 at each time step. This is Q^* optimal. f can be chosen such that $f((m_1, o_{\text{end}}), a) = 1$ and $f((m_2, o_{\text{end}}), a) = 0$ which gives an Q^* error of 0 for terminal trajectories. For non-terminal partial trajectories we note that the true future return is independent of the actual time step because there is no time dependence for transitions nor termination. This lets us define $\hat{R}(s_1)$ to be the future discounted rewards if the current MDP state is s_1 and $\hat{R}(s_2)$ to be the future discounted rewards if the current MDP state is s_2. We can then choose $f((m_1, o_x \neq o_{\text{end}}), a) = \hat{R}(s_1)$ and $f((m_2, o_x \neq o_{\text{end}}), a) = \hat{R}(s_2)$ which also gives $\epsilon_{Q^*} = 0$. This is because $P(m_1|\tau) = P(s_1|\tau)$ and $P(m_2|\tau) = P(s_2|\tau)$ so when lifting for a given trajectory τ we get $P(m_1) * f((m_1, o \neq o_{\text{end}}), a) + P(m_2) * f((m_2, o \neq o_{\text{end}}), a) = P(s_1) * \hat{R}(s_1) + P(s_2) * \hat{R}(s_2)$ which is exactly the true future discounted reward.

Following similar reasoning, this memory function is also Model optimal. For terminal trajectories f_R can match f and for non-terminal trajectories $f_R((\cdot, o \neq o_{\text{end}}), a) = 0$ which gives $\epsilon_R = 0$. For transitions, the probability of o_{end} is always .1 and the probability of the observations o_x follows $U(-1, 1) * .9$ which gives a natural choice of f_P with $\epsilon_P = 0$.

No memory can at best achieve $\epsilon_{Q^*} = 1/2$ and $\epsilon_R = 1/2$ because the true reward at the final observation can be either 0 or 1 and $f(o_{\text{end}}, a)$ can at best be assigned to the middle of this range to minimize error.

We now consider a k-deterministic memory function μ, with corresponding . Lets assume that exists function $f : \phi(T) \times A \to \mathbb{R}$ such that $||f(\phi(\tau)) - R(\tau)||_\infty \leq \epsilon < 1/2$. Note that this is identical to the terminal trajectory requirement for f in Lemma 4.2. For simplicity we can exclude the observation and action, which are always o_{end} and a respectively, to get an identical $f : M \to \mathbb{R}$. We now prove by contradiction that such f cannot exist.

For each memory state m_i we define $S_i = \{\tau | \phi(\tau) = m_i, \tau \text{ is terminal}\}$ and $R(S) = \{R(\tau) | \tau \in S\}$. Let S, generated by memory state m, be the set for which $\sup R(S) - \inf R(S) \geq \sup R(S_i) - \inf R(S_i)$ for all S_i. The best choice of $f(m)$ is $(\sup R(S) + \inf R(S))/2$ because for all $\tau \in S$, $|f(m) - R(\tau)| \leq (\sup R(S) - \inf R(S))/2 \leq \epsilon < 1/2$. This implies that $\sup R(S) - \inf R(S) \leq 2\epsilon < 1$. Either $\inf S > 0$ or $\sup S < 1$. Without loss of generality, assume that $\inf S > 0$.

We now consider an arbitrary trajectory τ and define the operation $\tau \oplus o_x$ for observation o_x which generates a new terminal trajectory by inserting the observation o_x before o_{end} in the trajectory. Notice that for any $\tau_1, \tau_2 \in S$ and o_x, we have that $\phi(\tau_1 \oplus o_x) = \phi(\tau_2 \oplus o_x) = \mu(m, a, o_x)$. We also have that for positive x, $R(\tau \oplus o_x) = (1-x)R(\tau)$ as defined by the probability of transitioning from s_1 to s_1 in the virtual MDP.

For any choice $0 < \epsilon' < \frac{1}{4} \sup R$ we can choose $\tau_1, \tau_2 \in S$ and o_x such that $\sup R(S) - R(\tau_2 \oplus o_x) = \epsilon'$ and $R(\tau_1 \oplus o_x) < \inf R(S)$. First we pick $\tau_2 \in S$ such that $\sup R(S) - R(\tau_2) = \delta < \epsilon'$, for $\delta \in \mathbb{R}$, which gives $R(\tau_2) = \sup R(S) - \delta$. This then means we can choose $x = 1 - (\sup R(S) - \epsilon')/(\sup R(S) - \delta)$ which gives the desired $\sup R(S) - R(\tau_2 \oplus o_x) = \epsilon'$. The condition $0 < \epsilon' < \frac{1}{4} \sup R$ ensures $x \in (0, 1]$. We can now choose $\tau_1 \in S$ such that $R(\tau_1) - \inf R(S) = \delta' < \frac{x}{1-x} \inf R(S)$ which reduces to the desired $R(\tau_1)(1 - x) < \inf R(S)$ which is equivalent to $R(\tau_1 \oplus o_x) < \inf R(S)$.

We now consider a sequence of choices of $\epsilon', \epsilon_1, \epsilon_2, \ldots$, such that $\epsilon_i = \epsilon_{i-1}/2$. For each choice of epsilon ϵ_i we have $\tau_{1,i}, \tau_{2,i} \in S$ and o_x such that $\sup R(S) - R(\tau_{2,i} \oplus o_x) = \epsilon'$ and $R(\tau_{1,i} \oplus o_x) < \inf R(S)$. Let $m_i = \phi(\tau_{1,i} \oplus o_x) = \phi(\tau_{2,i} \oplus o_x)$ for each ϵ_i. For the infinite sequence of m_i, there must be some particular $\hat{m}$ that repeats infinitely many times. Let $\hat{m}$ generate $\hat{S}$ and I be the set of $\{i | m_i = \hat{m}\}$. For the pairs $\tau_{1,i} \oplus o_x, \tau_{2,i} \oplus o_x \in \hat{S}$, we have that $\inf R(\hat{S}) \leq \inf_{i \in I} R(\tau_{1,i} \oplus o_x) < \inf R(S)$ and $\sup_{i \in I} R(\tau_{2,i} \oplus o_x) = \sup R(S) \leq \sup R(\hat{S})$. This implies that $\sup R(\hat{S}) - \inf R(\hat{S}) > \sup R(S) - \inf R(S)$ which contradicts the definition of S. So we have that $||f(\phi(\tau)) - R(\tau)||_\infty \geq 1/2$ and by Lemma 4.2 we have that $\epsilon_{Q^*} \geq 1/2$ and $\epsilon_M \geq 1/2$.

4.1.5 Stochastic Memory Can't Count Multiples - Model and Q^*

Our final counter example shows a scenario where k deterministic memory function can perfectly predict the return and model a POMDP while a 2 state stochastic memory cannot. This highlights that while normally stochastic memory transitions are often much more expressive, there are scenarios where the

exact precision of deterministic memory is necessary.

Consider an environment where the agent needs to keep track of the multiplicity of the time step. First, the agent receives a sequence of 0, 1, 2, or 3 o_{null} observations followed by a single o_{end} observation. At each time step, the agent can only take the action a. After each observation, the agent receives a reward of 0 unless the observation is o_{end} and the time step is a multiple of 3. More specifically, here are the rewards for the following observation sequences:

1. $R(o_{\text{end}}) = 1$
2. $R(o_{\text{null}}, o_{\text{end}}) = 0$
3. $R(o_{\text{null}}, o_{\text{null}}, o_{\text{end}}) = 0$
4. $R(o_{\text{null}}, o_{\text{null}}, o_{\text{null}}, o_{\text{end}}) = 1$

Each of the possible trajectory sequences is equally likely.

A 3-state deterministic memory function is sufficient to achieve 0 reward error in this environment. Consider three states m_1, m_2, and m_3 that transition in a cycle with 100% probability. Whenever the memory is in state m_1, the initial memory state, the agent can predict a reward of 1 and otherwise predict a reward of 0. This gives a candidate f which satisfies $||[f]_\phi - Q^*_M||_\infty \leq \epsilon_{Q^*}$.

An agent with no memory could achieve a maximum error of $1/2$ by predicting a reward of $1/2$ in all cases.

We now consider the constraints that a 2-state stochastic memory function would need to satisfy in order to perform better than an agent with no memory. Note that because the agent only has one available action, we can think of f as a function only of memory. We will also reduce our view to only the o_{end} observation because if no f exists which outperforms a memoryless agent over just one of the observations it also cannot exist over both. Because we are considering only a single action and a single observation, f becomes a function of only the memory state. This lets us succinctly express $[f]_\phi$ as $\mathbb{E}_{(m,o)\sim\phi(\tau)}[f(m)] = \Pr(m_1|\tau)f(m_1) + \Pr(m_2|\tau)f(m_2)$. For convenience, we write $f = (f(m_1), f(m_2))$.

We now need to determine what $Pr(m_1)$ and $Pr(m_2)$ would be for a given trajectory. Because the observation is always o_{null} for all time steps before the o_{end} observation, and because the agents action is always a, the memory function update reduces to a function of only the previous memory state. This also lets us express it as a two by two matrix $A = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$, where p is the probability of transitioning to m_1 when in m_1 and q is the probability of transitioning to m_2 when in m_2. Now, given a vector representing the probabilities of each memory state, we can find the corresponding probabilities at the next time step by multiplying this vector by A. Finally, we say the initial memory state distribution is $y = (y_1, y_2)$ where y_1 is the probability of starting in state m_1 and y_2 is the probability of starting in state m_2.

Using y, A, and f, we can express the predicted Q^* value, which is equivalently the final reward, for the terminal states of the four possible trajectories

of this environment. If we assume that this 2-stochastic memory agent is improving, we know that these predictions must be greater or less than $1/2$ based on the true Q^* value.

1. $yf^T > 1/2$
2. $yAf^T < 1/2$
3. $yA^2f^T < 1/2$
4. $yA^3f^T > 1/2$

Note that these inequalities are strict because predicting $1/2$ would mean that the error of the agent is at least $|1/2 - 1|$ or $|1/2 - 0|$ which is not better than a no-memory agent.

From the first two conditions, we get that $yAf < yf$, and subtracting $yf = yIf$, where I is the identity matrix, we get $y(A - I)f < 0$. From the second two conditions we get that $yA^2f < yA^3f$ and subtracting yA^2f gives $0 < y(A^3 - A^2)f$. We can calculate $A^3 - A^2$ to be $(a + b - 1)^2(A - I)$ so we get the final condition of $0 < (a + b - 1)^2y(A - I)f$.

Because $(a + b - 1)^2$ is positive we have a contradiction. Both $0 < (a + b - 1)^2y(A - I)f$ and $y(A - I)f < 0$ cannot be true. This implies that a 2-stochastic memory agent cannot perform any better than a no memory agent on this environment.

4.2 The POMDP Hierarchy

In this section, I set out to define classes of POMDPs in terms of memory functions, to determine which classes include each other through our proofs and counter examples, and to build a structured POMDP class hierarchy. What our proofs and counter examples show is that the POMDP hierarchy I desired is actually quite sparse. Excluding trivial relations ships, the existence of one kind of memory tells you almost nothing [2] about the existence of any other kind of memory. While the various counter examples presented in this section don't produce a satisfying POMDP class hierarchy but, they do highlight the tremendous complexity of environments which are represented by the POMDP space.

The particular types of memory function considered here are by far not exhaustive and one could consider many more variations. On particularly limiting constraint is that I consider only memory functions with 2 and k memory states for arbitrary k. Focusing on specific integer values for memory size, for example, might help produce more granular memory function dependencies and could lead to a clearer POMDP class hierarchy.

[2] The notable exception being the proof given in section 4.1.2

5 Conclusion

In this work I present two separate views on agent memory in the context of POMDPs; trajectory sets and finite memory functions. Trajectory sets are shown to have several interesting theoretical properties. They preserve the Markov property and this allows them to be composed into an MDP. Trajectory sets also represent the information that a learning agent obtained so far and can be converted into an optimal memory function. Although these results are promising, I find trajectory sets to be non-practical due to their complexity. Further study would be necessary to determine if these drawbacks can be avoided and if trajectory sets can be used for learning memory functions.

In the second half of this work, I explore and construct a POMDP hierarchy in terms of different types of agent memory functions. To build the links in this hierarchy I present counter examples and proofs to show if the existence of one kind of memory function necessarily implies the existence of another. The results didn't lead to a satisfying hierarchy but many results are quite interesting on their own. In particular, I show that deterministic memory can closely approximate stochastic memory and that in specific POMDPs deterministic memory can be necessary.

6 acknowledgments

I would like to thank Dr. Konidaris for supervising my research work during my master's program. I would also like to thank Aaron Kirtland for his kind mentoring and helpful discussion of many of the ideas, proofs, and examples presented in this work.

References

[1] Daniel S Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes, 2013.

[2] Fan Chen, Huan Wang, Caiming Xiong, Song Mei, and Yu Bai. Lower bounds for learning in revealing pomdps, 2023.

[3] Jiacheng Guo, Minshuo Chen, Huan Wang, Caiming Xiong, Mengdi Wang, and Yu Bai. Sample-efficient learning of pomdps with multiple observations in hindsight, 2023.

[4] L. Li, T.J. Walsh, and M.L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, 2006.

[5] Long-Ji Lin and Tom M Mitchell. *Memory approaches to reinforcement learning in non-Markovian domains*. Citeseer, 1992.

[6] Michael Littman. Algorithms for sequential decision making. pages 120–128, 08 2009.

[7] Lisa Meeden, Gary McGraw, and Douglas Blank. Emergent control and planning in an autonomous vehicle. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 15, 1993.

[8] Mark Bishop Ring. *Continual learning in reinforcement environments*. The University of Texas at Austin, 1994.

[9] Jürgen Schmidhuber. Reinforcement learning in markovian and non-markovian environments. *Advances in neural information processing systems*, 3, 1990.

[10] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Lawson L.S. Wong, and Stefanie Tellex. Multi-object search using object-oriented pomdps. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7194–7200, 2019.