Radboud University Nijmegen

Faculty of Science

Modelling Non-Markovian Properties

-

Thesis MSc Computing Science

Author:
Serena Rietbergen

Supervisor:
dr. Nils Jansen

Second reader:
- -

-

Contents

Chapter 1

Introduction

Chapter 2

Belief-based Rewards

2.1 Partially Observable MDPs

Partially Observable Markov Decision Processes have been used to model problems concerning sequential decision-making where the model is uncertain. The objective is to act in a way that optimizes a performance criterion. This often translates itself to maximizing a certain reward, or minimizing certain costs involved.

2.1.1 Definitions

Definition 2.1 (Partially Observable Markov Decision Process)**.** A Partially Observable Markov Decision Process, or POMDP, is a tuple $(S, \mathcal{A}, \Omega, T, O, r, b_0)$, where

- S, the state space;
- A, the action space;
- Ω, the observation space;
- $T : S \times \mathcal{A} \times S \to [0, 1]$, the transition function where $T(s, a, s') = Pr(s' \mid s, a)$;
- $O : S \times \mathcal{A} \times \Omega \to [0, 1]$, the observation function where $O(s', a, o) = Pr(o \mid s', a)$
- $r : S \times \mathcal{A} \to \mathbb{R}$, the reward function, and
- $b_0 : S \to [0, 1]$, the initial probability distribution over states.

A POMDP is only partially observable, which indicates that we do not know everything about the model. We specifically are unaware of what state a POMDP currently is in. To create a notion of what state we believe to be in, we use belief states.

Definition 2.2 (Belief state)**.** Given a POMDP $(S, \mathcal{A}, \Omega, T, O, r, b_0)$, a belief state b is a probability distribution over S called Δ. The probability of the POMDP being in state s according to belief b, is denoted by $b(s)$. Logically $\sum_{s \in S} b(s) = 1$ and for all states s we have that $b(s) \in [0, 1]$.

A belief state evolves over time and a POMDP will start in their initial belief state b_0. The belief state will be updates according to the action that is undertaken and an observation that will be observed after taking said action.

Definition 2.3 (Updated belief state)**.** Let $Pr(o|a,b) = \sum_{s,s' \in S} O(s'', a, o)T(s, a, s'')b(s)$. The probability of ending in state s' after performing an action a and then observing observation o is calculated as

$$b^{a,o}(s') = \frac{O(s,a,o)}{Pr(o|a,b)} \sum_{s \in S} T(s,a,s')b(s)$$

With the help of belief states, we can create a fully observable continuous space belief MDP representing a POMDP.

Definition 2.4 (Belief MDP)**.** A POMDP $(S, A, \Omega, T, O, r, b_0)$ can be written as a belief MDP $(\Delta, \mathcal{A}, \tau, \rho)$, where

- $\Delta = \Pi(S)$, the set of probability distributions over S;
- $\mathcal{A}$, the action space;
- $\tau : \Delta \times \mathcal{A} \times \Delta \to [0, 1]$, the new transition function
- $\rho : \Delta \times \mathcal{A} \to \mathbb{R}$, the new reward function.

The objective is to maximize the expected cumulative reward, which is done by looking for an optimal policy. Let γ be some discount factor and ρ_t be the expected immediate reward obtained at a certain step t, then the expected cumulative reward is defined as

$$J^{\pi}(b) = E\big[\sum_{t=0}^{\infty} \gamma \rho_t \mid b, \pi\big]$$

With this information, we can calculate the optimal policy

$$\pi^* = \arg\max_{\pi \in \mathcal{A}^{\Delta}} J^{\pi}(b_0) \tag{2.1}$$

The value $J^{\pi^*}(b_0)$ can be computed recursively through a value function V_n, due to Bellman's principle of optimality[2]. If we let H be the possible infinite horizon of the problem, the optimal value can be concluded from $J^{\pi^*}(b) = V_{n=H}(b)$.

Definition 2.5 (Value function for POMDPs)**.** For $b \in \Delta$, let the value function be defined as

$$\begin{aligned} V_0(b) &= 0 \\ V_n(b) &= \max_{a \in \mathcal{A}} \big[\rho(b,a) + \gamma \int_{\Delta} \tau(b,a,b') V_{n-1}(b') db'\big] \\ &= \max_{a \in \mathcal{A}} \big[\rho(b,a) + \gamma \sum_{o \in \Omega} Pr(o \mid a, b) V_{n-1}(b^{a,o})\big] \end{aligned} \tag{2.2}$$

Observe that the reward function for a POMDP is based on a state and action through $r(s, a)$, while the reward function for a belief MDP depends on the belief state and action through $\rho(b, a)$. The belief MDP reward function ρ can be derived from the POMDP reward function r like so

$$\rho(b,a) = \sum_{s \in S} b(s) r(s,a) \tag{2.3}$$

When the ρ is defined as in Equation 2.3, the recursive computation in Equation 2.2 has the property to generate piecewise-linear and convex (PWLC) value functions for each horizon[3].

Definition 2.6 (Piecewise-linear function)**.** A function is piecewise-linear if it consists of n linear segments defined over n intervals.

Definition 2.7 (Convex function)**.** A function $f : A \to B$ is convex if for all $\theta \in [0, 1]$ and $x_1, x_2 \in A$

$$f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2)$$

For example, if we claim that Γ_n is the set of vectors representing the value function for horizon n, we can rewrite the value function to $V_n(b) = \max_{\alpha \in \Gamma_n} \sum_{s \in S} b(s)\alpha(s)$.

2.1.2 Solving with exact updates

Using the PWLC property, Equation 2.2 can be refactored to be able to perform the Bellman update. Let $\chi_n(b, s) = (\chi_n(b))(s)$ where $\chi_n(b) = \arg\max_{\alpha \in \Gamma_n} b \cdot \alpha$, so $\chi_n(b)$ returns the hyperplane that is maximal for a given belief state b. The refactored value function is then written as

$$V_n(b) = \max_{a \in \mathcal{A}} \sum_{o \in \Omega} \sum_{s \in S} b(s)\big[\frac{r(s, a)}{|\Omega|} + \sum_{s' \in S} T(s, a, s')O(s', a, o)\chi_{n-1}(b^{a,o}, s')\big] \tag{2.4}$$

Let's solely look at the terms within in the brackets. They are dependent on a certain action and observation. So for a given $o \in \Omega, a \in \mathcal{A}$, the set of α-vectors is generated by $\overline{\Gamma_n}^{a,o}$. Let $r^a(s) = r(s, a)$ and $P^{a,o}(s, s') = T(s, a, s')O(s', a, o)$ in

$$\overline{\Gamma_n}^{a,o} = \{\frac{r^a}{|\Omega|} + P^{a,o} \cdot \alpha_{n-1} \mid \alpha_{n-1} \in \Gamma_{n-1}\} \tag{2.5}$$

Observe that there are $|\mathcal{A}| \times |\Omega|$ of such $\overline{\Gamma_n}^{a,o}$ sets. To create the set containing all of these possible observation and action combination we take the following cross-sum:

$$\overline{\Gamma_n} = \bigcup_{a \in \mathcal{A}} \bigoplus_{o \in \Omega} \overline{\Gamma_n}^{a,o} \tag{2.6}$$

Note that the operator $\oplus$ is defined as $U \oplus W = \{u + w | u \in U, w \in W\}$.

Since $\overline{\Gamma_n}$ is created through all possible combination, it also contains α-vectors that are useless because the corresponding hyperplanes are below the value function. To create the actual Γ_n set from $\overline{\Gamma_n}$, we can use one of several pruning techniques on the $\overline{\Gamma_n}$ set which will get rid of the dominating useless vectors.

2.2 Extension for Active Sensing

All problems who only have partial observability tackle a certain issue through getting information about the model or achieving some goal. Most of the time, the concerning problem is tackled through obtaining enough information get an optimal reward. However, some *active sensing* problems (where the objective is acting in a way to acquire knowledge about a part of the model or minimizing the entropy over a given state variable) cannot be resolved this way. For these problems, the performance of an agent should incorporate their obtained knowledge of the system thus far.

In Equation 2.3, we observe that the reward of a POMDP is solely based on the current state and action. We want the agent to include their obtained knowledge, but simply putting the entire history of the model in the state, would lead to a combinatorial

explosion.

To tackle this issue, we propose a new method of defining rewards based on the acquired knowledge through the usage of belief states. Since we will use a different definition than the one described in Equation 2.3, the direct link with POMDPs is broken. Luckily, a lot of the components of a POMDP are still usable. For this we use a generalization of the POMDP framework, where the reward is directly defined as a function $\rho(b, a)$.

2.2.1 Definitions

Definition 2.8 (ρPOMDP)**.** A $\rho-$based POMDP (ρPOMDP) is a tuple $(S, \mathcal{A}, T, O, \rho, b_0)$, where

- S, the state space;
- A, the action space;
- Ω, the observation space;
- $T : S \times \mathcal{A} \times S \to [0, 1]$, the transition function where $T(s, a, s') = Pr(s' \mid s, a)$;
- $O : S \times \mathcal{A} \times \Omega \to [0, 1]$, the observation function where $O(s', a, o) = Pr(o \mid s', a)$;
- $\rho : \Delta \times \mathcal{A} \to \mathbb{R}$, the reward function, and ;
- $b_0 : S \to [0, 1]$, the initial probability distribution over states.

We see that ρ is often just some uncertainty or error measure, but it can also be combined with some expected state-reward.

2.2.2 Convexity property

Observe that the belief-based value function for POMDPs is convex (Equation 2.2), because $r(s, a)$ is linear in Equation 2.3. The value function for ρ-POMDPs is also convex, if $\rho(b, a)$ is convex.

Theorem 2.9. *If ρ and V_0 are convex functions over δ, then the value function V_n of the belief MDP is convex over Δ at any time step n.*

Note that $\rho(b, a)$ being convex over Δ is due to the fact that uncertainty measures are naturally convex[1]. We will only be focusing on reward functions that are convex. The initial value function $V_0 = 0$ for finite-horizon problems, but can be any convex function for infinite-horizon problems.

2.2.3 Solving with exact updates

We assume that this belief-based ρ is a PWLC function. Therefore it can be represented as several Γ-sets; one Γ_ρ^a for each action in $\mathcal{A}$:

$$\rho(b, a) = \max_{\alpha \in \Gamma_\rho^a} \big[\sum_s b(s)\alpha(s) \big] \tag{2.7}$$

We now rewrite the value function from Equation 2.2 together with the new reward function from Equation 2.7. Let $\chi_\rho^a(b, s) = (\chi_\rho^a(b))(s)$ where $\chi_\rho^a(b) = \arg\max_{\alpha \in \Gamma_\rho^a}(b \cdot \alpha)$. The rewritten value function for ρPOMDPs is then

$$V_n(b) = \max_{a \in \mathcal{A}} \sum_{s \in S} b(s) \big[\chi_\rho^a(b, s) + \sum_{o \in \Omega} \sum_{s' \in S} T(s, a, s') O(s', a, o) \chi_{n-1}(b^{a,o}, s') \big] \tag{2.8}$$

This new value function makes use of Γ_ρ^a through χ_ρ^a, but also generates $|\mathcal{A}| \times |\Omega|$ Γ-sets called $\overline{\Gamma_n}^{a,o}$, where $P^{a,o}(s, s') = T(s, a, s')O(s', a, o)$, defined as:

$$\overline{\Gamma_n}^{a,o} = \{P^{a,o} \cdot \alpha_{n-1} \mid \alpha_{n-1} \in \Gamma_{n-1}\} \tag{2.9}$$

Just like for POMDPs, we can use exact algorithms for ρPOMDPs. The difference is that now the cross-sum to calculate $\overline{\Gamma_n}$ now includes

- one $\alpha^{a,o}$ for each observation Γ-set $\overline{\Gamma_n}^{a,o}$, and
- one α_ρ from the Γ-set Γ_ρ^a corresponding to the reward.

This results in the new cross-sum

$$\overline{\Gamma_n} = \bigcup_{a \in \mathcal{A}} [\bigoplus_{o \in \Omega} \overline{\Gamma_n}^{a,o} \oplus \Gamma_\rho^a] \tag{2.10}$$

The number of vectors that are being generated is larger than for a regular POMDP. Let $|R|$ be the number of α-vectors specifying the $\rho(b, a)$ function, which depends on the action that was taken. The total amount of vectors is thus $|R| \times |\mathcal{A}||\Omega|$.

Chapter 3

Finite State Controllers

3.1 Definitions

Definition 3.1 (Parametric MDP)**.** $M = (S, s_I, \mathcal{A}, V, \mathcal{P})$ where

- S, the state space;
- s_I, the initial state;
- $\mathcal{A}$, the action space;
- V, the parameter space;
- $\mathcal{P} : S \times \mathcal{A} \times S \to \mathbb{Q}[V]$, the transition function.

  TO WRITE: polynomial notation and stuff

Let $A(s)$ denote the available actions, i.e. $A(s) = \{a \in \mathcal{A} \mid \exists s' \in S : \mathcal{P}(s, a, s') \neq 0\}$.

Definition 3.2 (MDP)**.** A MDP is a pMDP where

- $\mathcal{P} : S \times \mathcal{A} \times S \to [0, 1] \subseteq \mathbb{R}$
- for all states $s \in S, a \in \mathcal{A}, \sum_{s' \in S} \mathcal{P}(s, a, s') = 1$

Definition 3.3 (pMC)**.** A (parametric) Markov Chain is a (parametric) MDP where $|A(s)| = 1$ for all $s \in S$. Usually denoted as $D = (S, s_I, V, \mathcal{P})$, where $\mathcal{P} : S \times S \to \mathbb{Q}[V]$ because the actions are irrelevant.

Definition 3.4 (Strategy)**.** A strategy σ for a (p)MDP is a function that takes a finite path corresponding to M and returns a distribution over actions, $\sigma : \text{Paths}^M_{fin} \to \Pi(\mathcal{A})$. These strategies need to comply with $supp(\sigma(\pi)) \subseteq A(last(\pi))$.

TO WRITE: last, pi notation of paths, and supp

Definition 3.5 (POMDP)**.** A POMDP $\mathcal{M}$ can also be seen as a tuple (M, Ω, Obs) where

- $M = (S, s_I, \mathcal{A}, \mathcal{P})$, the underlying MDP;
- Ω, the finite set of observations;
- $Obs : S \to \Omega$, the observation function.

If for all states $s, s' \in S : Obs(s) = Obs(s')$, then $A(s) = A(s')$. If $O(s) = z$, then $A(z) = A(s)$.

Definition 3.6 (Observation-based Strategy)**.** An observation-based strategy is $\sigma : \text{Paths}^M_{fin} \to \Pi(\mathcal{A})$ for an underlying MDP M such that if for all paths $\pi, \pi' \in \text{Paths}^M_{fin}$ where $Obs(\pi) = Obs(\pi')$ we claim that $\omega(\pi) = \omega(\pi')$.

Definition 3.7 (Finite State Controller)**.** FSC $F = (N, n_I, \gamma, \delta)$ where

- N, the set of memory nodes
- n_I, the initial memory node
- $\gamma : N \times \Omega \to \Pi(\mathcal{A})$, the action mapping
- $\delta : N \times \Omega \to \mathcal{A} \to \Pi(N)$, the memory update.

3.2 Modelling MDPs with FSC

3.3 Modelling POMDPs with FSC

3.4 Belief-based Rewards

Chapter 4

Using Reward-FSC

Note that $\Pi(V)$ stands for a probability distribution over V. So $\Pi(V) = V \to [0,1]$ where $\sum_{v \in V} v = 1$.

Definition 4.1. A labeled Markov Decision Process (labeled MDP) is a tuple $M = (S, s_I, Act, T, \mathcal{P}, L, R_k)$ where

- S, the finite set of states;
- s_I, the initial state;
- Act, the finite set of actions;
- $T : S \times A \to \Pi(S)$, the probabilistic transition function;
- $\mathcal{P}$, a set of propositional values;
- $L : S \times Act \times S \to 2^{\mathcal{P}}$, the labeling function;
- $R_k : (S \times Act)^k \times S \to \mathbb{R}$, the history-based reward function.

It's sufficient to use labels to express a reward function [4].

Definition 4.2. A labeled partially observable MDP (labeled POMDP) is a tuple $\mathcal{M} = (M, Z, O, R_k)$ where

- $M = (S, s_I, Act, T, \mathcal{P}, L)$, the hidden MDP;
- Z, the finite set of observations;
- $O : S \to Z$, the observation function;

Based on the Non-Markovian reward function R_k, we build a finite-state controller that mimics their behavior.

Definition 4.3. A reward finite-state controller (R-FSC) for a reward function of a POMDP $\mathcal{M}$ is a tuple $\mathcal{F} = (N, n_I, \delta, \mathcal{R})$ where

- N, the finite set of memory nodes;
- n_I, the initial memory node;
- $\delta : N \times Z \times Act \to \Pi(N)$, the memory update;
- $\mathcal{R} : N \times Z \times Act \times N \to \mathbb{R}$, the reward update.

We use the notation $\delta(n, o, a, n')$ for the probability of ending up in the memory node n' starting in the memory node n, showing observation o and performing action a. Thus, $\delta(n, o, a, n') = \delta(n, o, a)(n')$.

When working with the trajectory $s_0 a_0 s_1 a_1 \dots s_{k+1} \in (S \times Act)^k \times S$ - which is a realization of the stochastic process performed a POMDP $\mathcal{M}$ - we also obtain two sequences from the connected R-FSC $\mathcal{F}$. First we obtain the memory sequence $n_0 n_1 \dots n_k$, where $n_0 = n_I$ and n_{i+1} is obtained with probability $\delta(n_i, O(n_i), a_i, n_{i+1})$. Second, we obtain the reward sequence $r_0 r_1 \dots r_k$. In this reward sequence we claim that $r_i = R(n_i, O(s_i), a_i, n_{i+1})$.

Since $\mathcal{F}$ simulates R_k of $\mathcal{M}$, we want the reward of R_k and of $\mathcal{F}$ to be the same when using the same trajectory. Thus $R_k(s_0 a_0 s_1 a_1 \dots s_{k+1}) = \mathcal{R}(n_k, O(s_k), a_k, n_{k+1}) = r_k$.

Since we have a labeling function, we can obtain a corresponding label sequence $l_0 l_1 \dots l_k \in 2^{\mathcal{P}}$ where $l_i = L(s_i, a_i, s_{i+1})$ following the same trajectory. These labels are an important part when constructing the reward machine that models the non-Markovian reward function R_k, since they form the input alphabet.

Definition 4.4. The induced Reward Machine for R-FSC on a labeled POMDP is a tuple $\mathcal{A} = (V, v_I, 2^{\mathcal{P}}, \mathbb{R}, \hat{\delta}, \sigma)$ where

- $V = S \times N$, the finite set of states;
- $v_I = (s_I, n_I)$, the initial state;
- $2^{\mathcal{P}}$, the input alphabet;
- $\mathbb{R}$, the output alphabet;
- $\hat{\delta} : V \times 2^{\mathcal{P}} \to V$, the transition function. We state $\hat{\delta}((s, n), l) = (s', n')$ if there are $a \in Act, s' \in S, n' \in N$ such that $L(s, a, s') = l$ and $\delta(n, O(s), a, n') > 0$.
- $\sigma : V \times 2^{\mathcal{P}} \to \mathbb{R}$, the output function. We state $\sigma((s, n), l) = \mathcal{R}(n, O(s), a, n')$ if there are $a \in Act, n' \in N$ such that $L(s, a, s') = l$ and $\delta(n, O(s), a, n') > 0$.

what if multiple actions allow for the specific label

what if multiple actions allow for the specific label

Bibliography

[1] Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 64–72. Curran Associates, Inc., 2010.

[2] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503 – 515, 1954.

[3] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[4] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 590–598. AAAI Press, 2020.