WIKIPEDIA

Partially observable Markov decision process

A **partially observable Markov decision process** (**POMDP**) is a generalization of a Markov decision process (MDP). A POMDP models an agent decision process in which it is assumed that the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state. Instead, it must maintain a probability distribution over the set of possible states, based on a set of observations and observation probabilities, and the underlying MDP.

The POMDP framework is general enough to model a variety of real-world sequential decision processes. Applications include robot navigation problems, machine maintenance, and planning under uncertainty in general. The general framework of Markov decision processes with imperfect information was described by Karl Johan Åström in 1965 [1] in the case of a discrete state space, and it was further studied in the operations research community where the acronym POMDP was coined. It was later adapted for problems in artificial intelligence and automated planning by Leslie P. Kaelbling and Michael L. Littman.[2]

An exact solution to a POMDP yields the optimal action for each possible belief over the world states. The optimal action maximizes (or minimizes) the expected reward (or cost) of the agent over a possibly infinite horizon. The sequence of optimal actions is known as the optimal policy of the agent for interacting with its environment.

Contents

Definition

Formal definition

A discrete-time POMDP models the relationship between an agent and its environment. Formally, a POMDP is a 7-tuple $(S, A, T, R, \Omega, O, \gamma)$, where

- S is a set of states,

- A is a set of actions,
- T is a set of conditional transition probabilities between states,
- $R : S \times A \to \mathbb{R}$ is the reward function.
- Ω is a set of observations,
- O is a set of conditional observation probabilities, and
- $\gamma \in [0, 1]$ is the discount factor.

At each time period, the environment is in some state $s \in S$. The agent takes an action $a \in A$, which causes the environment to transition to state s' with probability $T(s' \mid s, a)$. At the same time, the agent receives an observation $o \in \Omega$ which depends on the new state of the environment, s', and on the just taken action, a, with probability $O(o \mid s', a)$. Finally, the agent receives a reward r equal to $R(s, a)$. Then the process repeats. The goal is for the agent to choose actions at each time step that maximize its expected future discounted reward: $E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where r_t is the reward earned at time t. The discount factor γ determines how much immediate rewards are favored over more distant rewards. When $\gamma = 0$ the agent only cares about which action will yield the largest expected immediate reward; when $\gamma = 1$ the agent cares about maximizing the expected sum of future rewards.

Discussion

Because the agent does not directly observe the environment's state, the agent must make decisions under uncertainty of the true environment state. However, by interacting with the environment and receiving observations, the agent may update its belief in the true state by updating the probability distribution of the current state. A consequence of this property is that the optimal behavior may often include (information gathering) actions that are taken purely because they improve the agent's estimate of the current state, thereby allowing it to make better decisions in the future.

It is instructive to compare the above definition with the definition of a Markov decision process. An MDP does not include the observation set, because the agent always knows with certainty the environment's current state. Alternatively, an MDP can be reformulated as a POMDP by setting the observation set to be equal to the set of states and defining the observation conditional probabilities to deterministically select the observation that corresponds to the true state.

Belief update

After having taken the action a and observing o, an agent needs to update its belief in the state the environment may (or not) be in. Since the state is Markovian (by assumption), maintaining a belief over the states solely requires knowledge of the previous belief state, the action taken, and the current observation. The operation is denoted $b' = \tau(b, a, o)$. Below we describe how this belief update is computed.

After reaching s', the agent observes $o \in \Omega$ with probability $O(o \mid s', a)$. Let b be a probability distribution over the state space S. $b(s)$ denotes the probability that the environment is in state s. Given $b(s)$, then after taking action a and observing o,

$$b'(s') = \eta O(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)$$

where $\eta = 1/\Pr(o \mid b, a)$ is a normalizing constant with $\Pr(o \mid b, a) = \sum_{s' \in S} O(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)$.

Belief MDP

A Markovian belief state allows a POMDP to be formulated as a Markov decision process where every belief is a state. The resulting *belief MDP* will thus be defined on a continuous state space (even if the "originating" POMDP has a finite number of states: there are infinite belief states (in B) because there are an infinite number of probability distributions over the states (of S)).[2]

Formally, the belief MDP is defined as a tuple (B, A, τ, r, γ) where

- B is the set of belief states over the POMDP states,
- A is the same finite set of action as for the original POMDP,
- τ is the belief state transition function,
- $r : B \times A \to \mathbb{R}$ is the reward function on belief states,
- γ is the discount factor equal to the γ in the original POMDP.

Of these, τ and r need to be derived from the original POMDP. τ is

$$\tau(b, a, b') = \sum_{o \in \Omega} \Pr(b' | b, a, o) \Pr(o | a, b),$$

where $\Pr(o|a, b)$ is the value derived in the previous section and

$$Pr(b'|b, a, o) = \begin{cases} 1 & \text{if the belief update with arguments } b, a, o \text{ returns } b' \\ 0 & \text{otherwise} \end{cases}.$$

The belief MDP reward function (r) is the expected reward from the POMDP reward function over the belief state distribution:

$$r(b, a) = \sum_{s \in S} b(s) R(s, a).$$

The belief MDP is not partially observable anymore, since at any given time the agent knows its belief, and by extension the state of the belief MDP.

Policy and value function

Unlike the "originating" POMDP (where each action is available from only one state), in the corresponding Belief MDP all belief states allow all actions, since you (almost) always have *some* probability of believing you are in any (originating) state. As such, π specifies an action $a = \pi(b)$ for any belief b.

Here it is assumed the objective is to maximize the expected total discounted reward over an infinite horizon. When R defines a cost, the objective becomes the minimization of the expected cost.

The expected reward for policy π starting from belief b_0 is defined as

$$V^\pi(b_0) = \sum_{t=0}^{\infty} \gamma^t r(b_t, a_t) = \sum_{t=0}^{\infty} \gamma^t E\Big[R(s_t, a_t) \mid b_0, \pi\Big]$$

where $\gamma < 1$ is the discount factor. The optimal policy π^* is obtained by optimizing the long-term reward.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^\pi(b_0)$$

where b_0 is the initial belief.

The optimal policy, denoted by π^*, yields the highest expected reward value for each belief state, compactly represented by the optimal value function V^*. This value function is solution to the Bellman optimality equation:

$$V^*(b) = \max_{a \in A}\Big[r(b,a) + \gamma \sum_{o \in \Omega} \Pr(o \mid b, a) V^*(\tau(b,a,o))\Big]$$

For finite-horizon POMDPs, the optimal value function is piecewise-linear and convex.[3] It can be represented as a finite set of vectors. In the infinite-horizon formulation, a finite vector set can approximate V^* arbitrarily closely, whose shape remains convex. Value iteration applies dynamic programming update to gradually improve on the value until convergence to an ϵ-optimal value function, and preserves its piecewise linearity and convexity.[4] By improving the value, the policy is implicitly improved. Another dynamic programming technique called policy iteration explicitly represents and improves the policy instead.[5][6]

Planning in POMDP

Planning in POMDP is undecidable in general. However, some settings have been identified to be decidable (see Table 2 in,[7] reproduced below). Different objectives have been considered. Büchi objectives are defined by Büchi automata. Reachability is an example of a Büchi condition (for instance, reaching a good state in which all robots are home). coBüchi objectives correspond to traces that do not satisfy a given Büchi condition (for instance, not reaching a bad state in which some robot died). Parity objectives are defined via parity games; they enable to define complex objectives such that reaching a good state every 10 timesteps. The objective can be satisfied:

- almost-surely, that is the probability to satisfy the objective is 1;
- positive, that is the probability to satisfy the objective is strictly greater than 0;
- quantitative, that is the probability to satisfy the objective is greater than a given threshold.

We also consider the finite memory case in which the agent is a finite-state machine, and the general case in which the agent has an infinite memory.

Objectives	Almost-sure (infinite memory)	Almost-sure (finite memory)	Positive (inf. mem.)	Positive (finite mem.)	Quantitative (inf. mem)	Quantitative (finite mem.)
Büchi	EXPTIME-complete	EXPTIME-complete	undecidable	EXPTIME-complete[7]	undecidable	undecidable
coBüchi	undecidable	EXPTIME-complete[7]	EXPTIME-complete	EXPTIME-complete	undecidable	undecidable
parity	undecidable	EXPTIME-complete[7]	undecidable	EXPTIME-complete[7]	undecidable	undecidable

Approximate POMDP solutions

In practice, POMDPs are often computationally intractable to solve exactly, so computer scientists have developed methods that approximate solutions for POMDPs.[8]

Grid-based algorithms[9] comprise one approximate solution technique. In this approach, the value function is computed for a set of points in the belief space, and interpolation is used to determine the optimal action to take for other belief states that are encountered which are not in the set of grid points. More recent work makes use of sampling techniques, generalization techniques and exploitation of problem structure, and has extended POMDP solving into large domains with millions of states.[10][11] For example, adaptive grids and point-based methods sample random reachable belief points to constrain the planning to relevant areas in the belief space.[12][13] Dimensionality reduction using PCA has also been explored.[14]

Uses

POMDPs can be used to model many kinds of real-world problems. Notable applications include the use of a POMDP in management of patients with ischemic heart disease,[15] assistive technology for persons with dementia,[10][11] the conservation of the critically endangered and difficult to detect Sumatran tigers[16] and aircraft collision avoidance.[17]

References

1. Åström, K.J. (1965). "Optimal control of Markov processes with incomplete state information" (https://lup.lub.lu.se/record/8867084). *Journal of Mathematical Analysis and Applications*. **10**: 174–205. doi:10.1016/0022-247X(65)90154-X (https://doi.org/10.1016%2F0022-247X%2865%2990154-X).
2. Kaelbling, L.P., Littman, M.L., Cassandra, A.R. (1998). "Planning and acting in partially observable stochastic domains". *Artificial Intelligence*. **101** (1–2): 99–134. doi:10.1016/S0004-3702(98)00023-X (https://doi.org/10.1016%2FS0004-3702%2898%2900023-X).
3. Sondik, E.J. (1971). *The optimal control of partially observable Markov processes* (https://apps.dtic.mil/docs/citations/AD0730503) (PhD thesis). Stanford University.
4. Smallwood, R.D., Sondik, E.J. (1973). "The optimal control of partially observable Markov decision processes over a finite horizon". *Operations Research*. **21** (5): 1071–88. doi:10.1287/opre.21.5.1071 (https://doi.org/10.1287%2Fopre.21.5.1071).
5. Sondik, E.J. (1978). "The optimal control of partially observable Markov processes over the infinite horizon: discounted cost". *Operations Research*. **26** (2): 282–304. doi:10.1287/opre.26.2.282 (https://doi.org/10.1287%2Fopre.26.2.282).
6. Hansen, E. (1998). "Solving POMDPs by searching in policy space". *Proceedings of the Fourteenth International Conference on Uncertainty In Artificial Intelligence (UAI-98)*. arXiv:1301.7380 (https://arxiv.org/abs/1301.7380).
7. Chatterjee, Krishnendu; Chmelík, Martin; Tracol, Mathieu (2016-08-01). "What is decidable about partially observable Markov decision processes with ω-regular objectives" (https://doi.org/10.1016%2Fj.jcss.2016.02.009). *Journal of Computer and System Sciences*. **82** (5): 878–911. doi:10.1016/j.jcss.2016.02.009 (https://doi.org/10.1016%2Fj.jcss.2016.02.009). ISSN 0022-0000 (https://www.worldcat.org/issn/0022-0000).
8. Hauskrecht, M. (2000). "Value function approximations for partially observable Markov decision processes" (https://doi.org/10.1613%2Fjair.678). *Journal of Artificial Intelligence Research*. **13**: 33–94. doi:10.1613/jair.678 (https://doi.org/10.1613%2Fjair.678).

9. Lovejoy, W. (1991). "Computationally feasible bounds for partially observed Markov decision processes". *Operations Research*. **39**: 162–175. doi:10.1287/opre.39.1.162 (https://doi.org/10.1287%2Fopre.39.1.162).
10. Jesse Hoey; Axel von Bertoldi; Pascal Poupart; Alex Mihailidis (2007). "Assisting Persons with Dementia during Handwashing Using a Partially Observable Markov Decision Process". *Proc. International Conference on Computer Vision Systems (ICVS)*. doi:10.2390/biecoll-icvs2007-89 (https://doi.org/10.2390%2Fbiecoll-icvs2007-89).
11. Jesse Hoey; Pascal Poupart; Axel von Bertoldi; Tammy Craig; Craig Boutilier; Alex Mihailidis. (2010). "Automated Handwashing Assistance For Persons With Dementia Using Video and a Partially Observable Markov Decision Process". *Computer Vision and Image Understanding (CVIU)*. **114** (5): 503–519. CiteSeerX 10.1.1.160.8351 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.160.8351). doi:10.1016/j.cviu.2009.06.008 (https://doi.org/10.1016%2Fj.cviu.2009.06.008).
12. Pineau, J., Gordon, G., Thrun, S. (August 2003). "Point-based value iteration: An anytime algorithm for POMDPs" (http://www.fore.robot.cc/papers/Pineau03a.pdf) (PDF). *International Joint Conference on Artificial Intelligence (IJCAI). Acapulco, Mexico*. pp. 1025–32.
13. Hauskrecht, M. (1997). "Incremental methods for computing bounds in partially observable Markov decision processes". *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI). Providence, RI*. pp. 734–739. CiteSeerX 10.1.1.85.8303 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.85.8303).
14. Roy, Nicholas; Gordon, Geoffrey (2003). "Exponential Family PCA for Belief Compression in POMDPs" (http://papers.nips.cc/paper/2319-exponential-family-pca-for-belief-compression-in-pomdps.pdf) (PDF). *Advances in Neural Information Processing Systems*.
15. Hauskrecht, M. , Fraser, H. (2000). "Planning treatment of ischemic heart disease with partially observable Markov decision processes". *Artificial Intelligence in Medicine*. **18** (3): 221–244. doi:10.1016/S0933-3657(99)00042-1 (https://doi.org/10.1016%2FS0933-3657%2899%2900042-1). PMID 10675716 (https://pubmed.ncbi.nlm.nih.gov/10675716).
16. Chadès, I., McDonald-Madden, E., McCarthy, M.A., Wintle, B., Linkie, M., Possingham, H.P. (16 September 2008). "When to stop managing or surveying cryptic threatened species" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2544557). *Proc. Natl. Acad. Sci. U.S.A*. **105** (37): 13936–40. Bibcode:2008PNAS..10513936C (https://ui.adsabs.harvard.edu/abs/2008PNAS..10513936C). doi:10.1073/pnas.0805265105 (https://doi.org/10.1073%2Fpnas.0805265105). PMC 2544557 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2544557). PMID 18779594 (https://pubmed.ncbi.nlm.nih.gov/18779594).
17. Kochenderfer, Mykel J. (2015). "Optimized Airborne Collision Avoidance" (https://ieeexplore.ieee.org/document/7288641). *Decision Making Under Uncertainty*. The MIT Press.

External links

- Tony Cassandra's POMDP pages (http://www.cassandra.org/pomdp/index.shtml) with a tutorial, examples of problems modeled as POMDPs, and software for solving them.
- pomdp: Solver for Partially Observable Markov Decision Processes (POMDP) (https://cran.r-project.org/web/packages/pomdp/index.html) an R package providing an interface to Tony Cassandra's POMDP solver.
- zmdp (https://www.cs.cmu.edu/~trey/zmdp/), a POMDP solver by Trey Smith
- APPL (http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.HomePage), a fast point-based POMDP solver
- SPUDD (http://www.cs.uwaterloo.ca/~jhoey/research/spudd/), a factored structured (PO)MDP solver that uses algebraic decision diagrams (ADDs).
- pyPOMDP (https://bitbucket.org/bami/pypomdp), a (PO)MDP toolbox (simulator, solver, learner, file reader) for Python by Oliver Stollmann and Bastian Migge

- Finite-state Controllers using Branch-and-Bound (http://www.cs.uwaterloo.ca/~mgrzes/code/IsoFreeBB/) An Exact POMDP Solver for Policies of a Bounded Size
- POMDPs.jl (https://github.com/JuliaPOMDP/POMDPs.jl), an interface for defining and solving MDPs and POMDPs in Julia with a variety of solvers.