POMDPs: Partically Observable Markov Decision Processes

Julia Academy: POMDPs.jl

Decision Making Under Uncertainty

What is a POMDP?

Definition: POMDP. A *Partially observable Markov decision process* (POMDP) is an MDP with *state uncertainty*—meaning we cannot know the *true* state, only a *belief* about the true state using *observations*.

- Formally, a POMDP is defined by the following:

Table: MDP Problem Formulation: $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma \rangle$

Variable	Description	`POMDPs` Interface
$\mathcal{S}$	State space	`POMDPs.states`
$\mathcal{A}$	Action space	`POMDPs.actions`
$\mathcal{O}$	Observation space	`POMDPs.observations`
$T(s' \mid s, a)$	Transition function	`POMDPs.transition`
$R(s, a)$	Reward function	`POMDPs.reward`
$O(o \mid s')$	Observation function	`POMDPs.observation`
$\gamma \in [0, 1]$	Discount factor	`POMDPs.discount`

Remember, a POMDP is a ***problem formulation*** and ***not an algorithm***.

How are POMDPs different than MDPs?

- A POMDP[2] is an MDP with *state uncertainty*

$$\text{MDP: } \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$$
$$\text{POMDP: } \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma \rangle$$

- The agent receives an *observation* of the current state rather than the true state (potentially imperfect observations)
- Using past observations, the agent builds a *belief* of their underlying state
  - Which can be represented by a probability distribution over true states

[2] "Partially observable" is key in understanding beliefs.

Example POMDP: Crying Baby Problem

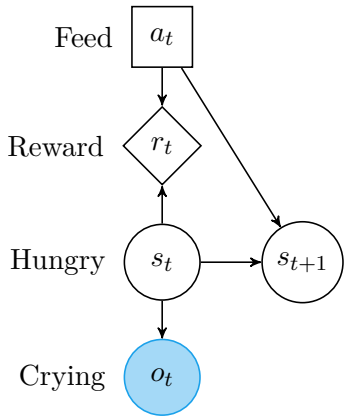


Figure: The crying baby POMDP.

- A simple POMDP with 2 states, 2 actions, and 2 observations:

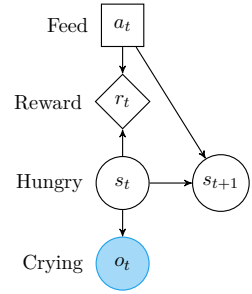
$$\mathcal{S} = \{\texttt{hungry}, \texttt{full}\}$$
$$\mathcal{A} = \{\texttt{feed}, \texttt{ignore}\}$$
$$\mathcal{O} = \{\texttt{crying}, \texttt{quiet}\}$$

- We cannot directly tell if the baby is truly `hungry`, but we can observe that it's `crying` and update our *belief* about the true state using this information.

QuickPOMDPs: Crying Baby

```julia
using POMDPs, POMDPModelTools, QuickPOMDPs

@enum State hungry full
@enum Action feed ignore
@enum Observation crying quiet

pomdp = QuickPOMDP(
    states      = [hungry, full],  # s
    actions     = [feed, ignore],  # a
    observations = [crying, quiet], # o
    initialstate = [full], # Deterministic
    discount    = 0.9, # γ

    transition = function T(s, a)
        if a == feed
            return SparseCat([hungry, full], [0, 1])
        elseif s == hungry && a == ignore
            return SparseCat([hungry, full], [1, 0])
        elseif s == full && a == ignore
            return SparseCat([hungry, full], [0.1, 0.9])
        end
    end,

    observation = function O(s, a, s′)
        if s′ == hungry
            return SparseCat([crying, quiet], [0.8, 0.2])
        elseif s′ == full
            return SparseCat([crying, quiet], [0.1, 0.9])
        end
    end,

    reward = (s,a)->(s == hungry ? -10 : 0) + (a == feed ? -5 : 0)
)
```

- This code[a] defines the entire *Crying Baby* POMDP using `QuickPOMDPs.jl`
  - Just a sneak-peek: we'll walk through this in detail in the `Pluto` notebooks



[a] Yes, this is self-contained—copy and paste it into a notebook or REPL!

POMDP SOLVERS

A number of ways to solve POMDPs are implemented in the following packages.

Table: POMDP Solution Methods

Package	Online/Offline	State Spaces	Actions Spaces	Observation Spaces
`QMDP.jl`	Offline	Discrete	Discrete	Discrete
`FIB.jl`	Offline	Discrete	Discrete	Discrete
`BeliefGridValueIteration.jl`	Offline	Discrete	Discrete	Discrete
`SARSOP.jl`	Offline	Discrete	Discrete	Discrete
`BasicPOMCP.jl`	Online	Continuous	Discrete	Discrete
`ARDESPOT.jl`	Online	Continuous	Discrete	Discrete
`MCVI.jl`	Offline	Continuous	Discrete	Continuous
`POMDPSolve.jl`	Offline	Discrete	Discrete	Discrete
`IncrementalPruning.jl`	Offline	Discrete	Discrete	Discrete
`POMCPOW.jl`	Online	Continuous	Continuous	Continuous
`AEMS.jl`	Online	Discrete	Discrete	Discrete
`PointBasedValueIteration.jl`	Offline	Discrete	Discrete	Discrete

When defining your problem, the ***type*** of state, action, and observation space is very important!