Tutorial 1: What is reinforcement learning?

Jacob Green

Abstract

We cover the Markov decision process formulation of reinforcement learning on games of complete information, and the partially observable Markov decision process formulation of reinforcement learning on games of incomplete information, showcasing the observation-level, episode-level and belief-level reward schemas.

1 Discrete Markov Decision Processes

The foundational work on Markov Decision Processes (MDPs) began with Bellman [1957], establishing the theoretical framework for dynamic programming and sequential decision-making. Howard [1960] further developed the connection between dynamic programming and Markov processes, providing the mathematical foundations that underpin modern reinforcement learning.

The most natural setting for reinforcement learning is a game. We will be working with discrete games (e.g. chess, poker), that have an *agent* interact with its *environment* at times $t = 0, 1, \ldots, N$. Each game comes with a set $\mathcal{S}$ containing all possible variants of the game's world, which we call the *state space*, and a set $\mathcal{A}$ containing all possible action-variants in the game's world, which we call the *action space*. For ease of notation, we will assume throughout $\mathcal{S}$ and $\mathcal{A}$ are discrete[1].

Example 1. *(simple chess) In the game of chess, the position of all the pieces on the chessboard describes the game at all times. Hence, we may let $\mathcal{S}$ be the space of 8×8 matrices with entries in $\{Pa, Bi, Kn, Ro, Qu, Ki\}$. For $\mathcal{A}$, we will assume no castling, en-passant, or pawn-upgrades to keep things simple. We can describe $\mathcal{A}$ by describing the possible moves for each piece, and assuming all pieces can be everywhere on the board. Under this assumption,*

$$\mathcal{A} = \boldsymbol{Pa} \cup \boldsymbol{Bi} \cup \boldsymbol{Kn} \cup \boldsymbol{Ro} \cup \boldsymbol{Qu} \cup \boldsymbol{Ki}$$

where

$$\boldsymbol{Pa} = \bigcup_{1 \leq i,j \leq 8} \{Pa_{i,j}((i, j+1), Pa_{i,j}(i, j+2)\}$$

$$\boldsymbol{Bi} = \bigcup_{1 \leq i,j,k \leq 8} \{Bi_{i,j}(i \pm k, j \pm k)\}$$

$$\boldsymbol{Kn} = \bigcup_{1 \leq i,j \leq 8} \{Kn_{i,j}(i \pm 1, j+2), Kn_{i,j}(i \pm 2, j+1), Kn_{i,j}(i \pm 1, j-1), Kn_{i,j}(i \pm 1, j-2)\}$$

$$\boldsymbol{Ro} = \bigcup_{1 \leq i,j,k \leq 8} \{Ro_{i,j}(i \pm k, j), Ro_{i,j}(i, j \pm k)\}$$

$$\boldsymbol{Qu} = \bigcup_{1 \leq i,j,k \leq 8} \{Qu_{i,j}(i \pm k, j), Qu_{i,j}(i \pm k, j \pm k), Qu_{i,j}(i, j \pm k)\}$$

$$\boldsymbol{Ki} = \bigcup_{1 \leq i,j \leq 8} \{Ki_{i,j}(i \pm 1, j), Ki_{i,j}(i \pm 1, j \pm 1), Ki_{i,j}(i, j \pm 1)\}$$

and for $P \in \{Pa, Bi, Kn, Ro, Qu, Ki\}$ we define $P_{i,j}(m, n)$ to be the action of moving piece P from coordinate (i, j) to (m, n).

[1] i.e. finite or countable

By describing our state space with a singular set[2] $\mathcal{S}$, and our action space with a singular set $\mathcal{A}$, we can make the following assumption.

Assumption 1. *(Games are Markovian) Throughout these tutorials, we will assume our games exhibit Markovian dynamics: i.e.*

$$\mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, s_{t-2}, \dots) = \mathbb{P}(s_{t+1}|s_t, a_t)$$

where $\mathbb{P}$ is a probability measure describing state transitions.

To have this Markov property, all the information gained in the game history $(s_0, a_0, s_1, \dots, s_{n-1}, a_{n-1}, s_n)$ up to time $t = n$ must be encoded in the state $s_n \in \mathcal{S}$. If this is the case, we can describe our game as a *discrete-time Markov process* by defining an *initial state distribution*

$$\mathbb{T}_0(s_0) : \mathcal{S} \to [0, 1]$$

and a *transition distribution*

$$\mathbb{T}(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$$

obeying, for all $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$\sum_{s' \in \mathcal{S}} \mathbb{T}_0(s') = 1 \quad \text{and} \quad \sum_{s' \in \mathcal{S}} \mathbb{T}(s'|s, a) = 1$$

Note, our game is only a discrete-time Markov process if all our actions are deterministic and pre-specified. As we will see, our framework is more general, though the intuition remains relevant.

Example 2. *(transition distributions for deterministic games) Suppose we have a deterministic game with state space $\mathcal{S}$ and action space $\mathcal{A}$. In this setting, we have an update function $f(a|s) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}, (s, a) \mapsto s$ that maps actions on a given state to updated states. We can specify the transition distribution for such a game, explicitly, by*

$$\mathbb{T}(f(a_t|s_t)|s_t, a_t) = 1$$

and $\mathbb{T} \equiv 0$ elsewhere.

[2] Note, we could not do this if we, for example, considered a "player state" that depended on the action history up to time n. Even if this space of player states was total, i.e. contained all possibilities on action histories, the next player state could depend on dynamics not contained in the previous state, action pair. We will consider these models, which we call POMDPS (partially observable Markov decision processes) next section, as a natural generalisation to our MDP (Markov decision process) model.

All together, we've found we can describe the dynamics of our game with the tuple $(\mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{T}_0)$. It remains to describe how our agent actually learns. The fundamental idea of reinforcement learning is as follows:

1. We **reinforce** desired behaviour by rewarding our agent
2. We **punish** undesired behaviour by penalising our agent

Recall example 1. In this example, given a current state, the vast majority of our action space $\mathcal{A}$ consists of invalid moves. We can have our agent *learn* to play only valid moves by *punishing* it when it attempts an invalid move. Likewise, we can have our agent *learn* to play good moves by *rewarding* it when it plays a good move, (e.g. taking a piece).

We will do this by considering *reward signals.* Given a state action pair (s, a) and a resulting state $s' \sim \mathbb{T}(\cdot|a, s)$ we choose some signal $r \in \mathbb{R}$ to give our agent, which contains the information on how desirable the outcome s' was. Our *reward signal map* is thus

$$R(s'|a, s) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$$

Is this by itself enough for our agent to learn? Consider the following example.

Example 3. *(infinite cumulative rewards) Consider the following game: At each time $t \geq 1$ you are given £1 and allowed the choice to pay £1 to bet on a fair coinflip. If the coin lands heads, you receive £10, and if the coin lands tails you lose your bet. This game has state space $\mathcal{S} = \{H, T\}$ and action space $\mathcal{A} = \{Bet, Fold\}$. The dynamics are given by*[3]

$$\mathbb{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1], (s'|a, s) = \frac{1}{2}$$

for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. The natural reward signal is simply the outcome of the bet plus 1, if we bet, and £1 otherwise. We have

$$R(s'|a, s) = \begin{cases} 11 & : \ s' = H \text{ and } a = \ Bet \\ 0 & : \ s' = T \text{ and } a = \ Bet \\ 1 & : \ a = \ Fold \end{cases}$$

We seek to find a strategy that maximises the expected cumulative reward for playing the game indefinitely,

$$\mathbb{E}_{s_1, s_2, \cdots \sim \mathbb{T}} \left[\sum_{t=0}^{\infty} R(s_{t+1}|a_t, s_t) \right]$$

[3] Note we do not need to specify an initial state distribution, nor consider the initial state in any meaningful way, as the transition distribution $\mathbb{T}(\cdot|a, s)$ is independent of s, so that $\mathbb{T}(\cdot|a, s) \equiv \mathbb{T}(\cdot|a)$. It is also independent of a.

Consider the case where we always bet, and the case where we never bet. In both cases

$$\mathbb{E}_{s_1,s_2,\dots\sim\mathbb{T}}\left[\sum_{t=0}^{\infty} R(s_{t+1}|a_t,s_t)\right]=\infty$$

as $R(s_{t+1}|a_t,s_t)\not\to 0$ *as* $t\to\infty$. *How can we decide which strategy is better?*

One approach is to consider the *expected discounted cumulative returns*

$$\mathbb{E}_{s_1,s_2,\dots\sim\mathbb{T}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}|a_t,s_t)\right]$$

for some fixed $\gamma\in[0,1)$. In this case, we see the never bet strategy has expected discounted cumulative return $\frac{1}{1-\gamma}$, whereas the always bet strategy has expected discounted return $\frac{5}{1-\gamma}>\frac{1}{1-\gamma}$. Hence, we can argue that the always bet strategy is indeed better than the never bet strategy.

Note that, by taking this approach, if R is a bounded function we will always have a finite expected discounted cumulative returns[4]. Sometimes, our model naturally should discount future returns (e.g. a stock-trading model would have to discount for the time-value of money), but other times, introducing this discount rate $\gamma\in[0,1)$ is simply a mathematical trick to ensure finite expected discounted cumulative returns on infinite horizons. If our horizon, N, is finite, we may take $\gamma=1$ and consider simply the expected cumulative returns, though this often is impractical.

All together, we've described a tuple $(\mathcal{S},\mathcal{A},\mathbb{T},\mathbb{T}_0,R,\gamma,N)$. We call this tuple a *discrete Markov decision process.* We conclude with a concise formal definition.

Definition 1. *(discrete MDP) A discrete Markov decision process is a tuple* $(\mathcal{S},\mathcal{A},\mathbb{T},\mathbb{T}_0,R,\gamma,N)$ *where*

- $\mathcal{S}$ *and* $\mathcal{A}$ *are sets, named the state and action spaces respectively.*
- $\mathbb{T}(s'|a,s):\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$ *is a conditional distribution, named the transition distribution, describing the dynamics of state-action updates.*
- $\mathbb{T}(s):\mathcal{S}\to[0,1]$ *is a distribution, named initial state distribution, describing the dynamics of initial state selection.*
- $R(s'|a,s):\mathcal{S}\times\mathcal{A}\times\mathcal{S}\to\mathbb{R}$ *is a map defining the reward signalled by observing state* s' *from the state-action pair* (s,a)

[4] $\exists M>0$ s.t. $\forall s',s\in\mathcal{S},a\in\mathcal{A}:|R(s'|s,a)|\le M\Rightarrow\sum_{t\geq 0}\gamma^t R(s_{t+1}|s_t,a_t)\le M\sum_{t=0}^{\infty}\gamma^t=\frac{M}{1-\gamma}<\infty$

- $\gamma \in [0,1)$ *denotes the discount rate*
- $N \in \mathbb{N} \cup \{+\infty\}$ *denotes the, possibly infinite, time horizon.*

If $N < +\infty$ *we will call our discrete Markov decision process finite.*

For the remainder of this section, let $(\mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{T}_0, R, \gamma, N)$ be a discrete MDP.

In the continuation of example 3, we compared two strategies by computing the expected discounted cumulative return of each strategy. We can abstract this approach to formalise the reinforcement learning objective. First, we need to define this notion of a strategy in abstract. The naive idea is to consider a map $\mathcal{S} \to \mathcal{A}$, i.e. for each state $s \in \mathcal{S}$, we specify a chosen action $a \in \mathcal{A}$. This is reasonable, but it fails in two regards:

1. We will never learn optimal strategies where action randomisation is optimal[5].
2. Given a learnt map $\mathcal{S} \to \mathcal{A}$, we have no uncertainty quantification.

Note the somewhat conflicting nature of these remarks. If randomisation is optimal, it may appear that we are just uncertain of our actions, and conversely if we are randomising, it may appear like a stochastic strategy, when in reality our agent is simply uncertain of its actions. It is non-trivial, in practice, to differentiate between the two. We'll provide a modern[6] approach next tutorial.

Given we'd like to allow for randomisation and uncertainty quantification, the natural approach is to instead consider a distribution[7] on $\mathcal{A}$ for each $s \in \mathcal{S}$. Formally, we define a *policy* as a map

$$\pi(a|s) : \mathcal{A} \times \mathcal{S} \to [0,1] \quad \text{with} \quad \forall s \in \mathcal{S} \ : \ \sum_{a \in \mathcal{A}} \pi(a|s) = 1$$

Given a an agent utilising a policy π for the duration of a game with horizon N, we will receive, upon the game's completion, $\tau = (s_0, a_0, s_1, r_1, a_1, \ldots, s_{N-1}, r_{N-1}, a_{N-1}, s_N, r_N)$ where $a_n \sim \pi(\cdot|s_n)$ and $r_n := R(s_{n+1}|s_n, a_n)$ for $n = 0, 1, \ldots, N-1$. We call τ an epsiode. If $N = +\infty$, then τ is an infinite tuple. If N is finite, note we have one more state observation than action/reward observation.

We'd like to know the distribution of τ. It can be proven (exercise) that, given a policy π, episodes are distributed by

$$\mathbb{T}(\tau; \pi) = \mathbb{T}_0(s_0) \prod_{t=1}^{N} \pi(a_t|s_t) \mathbb{T}(s_t|s_{t-1}, a_{t-1})$$

[5] See, for example, game theoretically optimal poker.

[6] It's pretty intuitive too, so we can motivate it.

[7] Or, equivalently, a conditional distribution on $\mathcal{A} \times \mathcal{S}$.

The goal of reinforcement learning is to learn π that maximises our *expected episodic discounted cumulative reward*

$$J(\pi\,;\,\mathbb{T}) = \mathbb{E}_{\tau\sim\mathbb{T}(\cdot\,;\,\pi)}\left[\sum_{t=1}^{N}\gamma^t R(s_{t+1}|s_t,a_t)\right]$$

Namely, we seek to solve the optimisation problem

$$\operatorname*{argmin}_{\pi} J(\pi;\mathbb{T})$$

over a specified space $\mathcal{P}$ of policies.

2 Partially Observable Discrete Markov Decision Processes

The extension to partially observable settings was pioneered by Smallwood and Sondik [1973], who established the theoretical foundations for optimal control under partial observability. The modern computational framework is due to Kaelbling et al. [1998], who provided algorithms for planning and acting in partially observable stochastic domains.

In many games, while the underlying dynamics are still given by a Markov process $(\mathcal{S},\mathcal{A},\mathbb{T},\mathbb{T}_0,N)$, we may only observe a fraction of the global state. The natural examples here are games of incomplete information, such as poker, and in robotics, where our robot can only "see" a fraction of the world it acts in. Let $\mathcal{O}$ be the set of possible observations. In poker, $\mathcal{O}$ would be the set of our possible hands, along with common hands, the hand's betting history metadata[8]. In robotics, it'd be the set of possible images we observe, e.g. the set of $H\times W\times 3$ tensors for $H\times W$ rgb images.

Our observation $o\in\mathcal{S}$ depends on the state. Naively, we could define a map $\mathcal{S}\to\mathcal{O}$ to describe the observation process, but this is subject to a critical fault: we cannot model observation noise. Consider the game often called "options trading". Our agent will be fed order-books for various options contracts, however, due to the nature of financial data[9], these are subject to some noise. What we observe may not, directly, be a function of the world, but some function of the world pertubed by noise. Hence, we define an *observation distribution* $\mathbb{O}(o|s):\mathcal{O}\times\mathcal{S}\to[0,1]$. This a conditional distribution, so for each $s\in\mathcal{S}$ we have $\sum_{o\in\mathcal{O}}\mathbb{O}(o|s)=1$.

Thus far, we have a tuple $(\mathcal{S},\mathcal{A},\mathcal{O},\mathbb{T},\mathbb{T}_0,\mathbb{O},N)$ describing the dynamics of the game and the observation process. It remains to define rewards, how our agent will learn, in this more general

[8] e.g. stack sizes, button position. We could also, if we want to train a reactive agent, include betting histories outside the the window of the current hand, along with all the usual observational data.

[9] Namely, latency compounded with continuous pricing.

environment. Last section, in the complete information case, we could define a reward signal $R(s'|a,s) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The naive approach for us is to instead consider a map $\mathcal{O} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$, a reward signal for observations. This, while certainly practical, makes a key assumption: observations are sufficiently informative to about rewards. More formally, we're assuming

$$\forall s_0, s_1, s_0', s_1' \in \mathcal{S}, a \in \mathcal{A} : (\mathbb{O}(o|s_0) > 0 \wedge \mathbb{O}(o|s_0') > 0) \implies R(s_1|a, s_0) = R(s_1'|a, s_0')$$

Consider the game of heads-up poker. If we observe our opponent folding to our bet, our natural reward $\mathcal{O} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$ would be the amount we won from the pot. This, however, misses a great deal of possible reward structure. A full $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ reward would factor in the cases where we bluffed our opponent off of a stronger hand than ours (greater reward, gained value) and the case our opponent has a weaker hand (lesser reward, missed value).

So, we'd still like to still use our reward signal $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The question then is, how do we compute $R(s'|a,s)$ if we only have access to $o, o' \in \mathcal{O}$? An approach often taken in the literature[10] is to maintain *beliefs* $b_t : \mathcal{S} \to [0,1]$, which is a distribution over $\mathcal{S}$ describing the agent's certainty we are in a state $s \in \mathcal{S}$, throughout the times $t = 0, 1, 2, \ldots, N$. We then approximate $R(s_{t+1}|a_t, s_t)$ with

$$\tilde{R}(b_{t+1}|a_t, b_t) = \mathbb{E}[R(s_{t+1}|a_t, s_t)|b_{t+1}, a_t, b_t]$$

We call this map $\tilde{R} : \Delta(\mathcal{S}) \times \mathcal{A} \times \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the space of distributions over $\mathcal{S}$, or equivalently the $\mathcal{S}$-simplex, our *belief-level reward signal*, which approximates the unobservable reward signal $R(s_{t+1}|a_t, s_t)$. This is an expectation over the joint **posterior distribution** for (s_t, s_{t+1}) given (b_t, a_t, b_{t+1}).

Proposition 1 (belief-level reward signal formula)**.** *Given the setup above, we have*

$$\tilde{R}(b_{t+1}|a_t, b_t) = \frac{\sum_{s_t, s_{t+1} \in \mathcal{S}} b_t(s_t) \mathbb{T}(s_{t+1}|s_t, a_t) \mathbb{O}(o_{t+1}|a_t, s_{t+1}) R(s_{t+1}|a_t, s_t)}{\sum_{s, s' \in \mathcal{S}} b_t(s) \mathbb{T}(s'|s, a_t) \mathbb{O}(o_{t+1}|a_t, s')}$$

Proof. Our beliefs at time $t+1$, b_{t+1} are obtained from our beliefs at time t, b_t, action a_t, and observation at time $t+1$, o_{t+1}, so we may express

$$\tilde{R}(b_{t+1}|a_t, b_t) = \mathbb{E}[R(s_{t+1}|a_t, s_t)|b_t, a_t, o_{t+1}] = \sum_{s_t, s_{t+1} \in \mathcal{S}} \mathbb{P}(s_t, s_{t+1}|b_t, a_t, o_{t+1}) R(s_{t+1}|a_t, s_t)$$

where $\mathbb{P}$ is the probability measure for the joint distribution (s_t, s_{t+1}). By Bayes' theorem[11].

$$\mathbb{P}(s_t, s_{t+1}|b_t, a_t, o_{t+1}) = \frac{\mathbb{O}(o_{t+1}|a_t, s_{t+1}) b_t(s_t) \mathbb{T}(s_{t+1}|s_t, a_t)}{\sum_{s, s' \in \mathcal{S}} b_t(s) \mathbb{T}(s'|s, a_t) \mathbb{O}(o_{t+1}|a_t, s')}$$

The result immediately follows. □

[10] More so the theoretical literature, though it's not too uncommon in the applied literature.

[11] Details are left to an exercise

Note, we haven't actually specified how we update our beliefs in practice. That is a topic for another tutorial[12], the idea behind exhibiting proposition 1 now was simply to show that, given a fully specified observation distribution $\mathbb{O}$, transition distribution $\mathbb{T}$, global reward signal R and beliefs b_t, we can explicitly compute the expected approximation. The idea behind introducing this approximation now was to show that, despite the impossibility of directly using R, to show that we can approximate it in various ways.

There is yet another approach to defining rewards: episodic. Rather than compute a signal at each time step t, we can instead compute an aggregated reward over each episode. In the POMDP setting without reward signals, it is clear our episodes will be of the form

$$\tau = (o_0, a_0, o_1, a_1, \ldots, o_{N-1}, a_{N-1}, o_N) \in (\mathcal{O} \times \mathcal{A})^N \times \mathcal{O} =: \mathcal{T}$$

Hence, we could define R as a map $\mathcal{T} \to \mathbb{R}$, the pre-specified episodic reward. For example, in a game where each oservation is a real, and our goal is to maximise the sum of our observed reals, we could take $R : \tau \mapsto \sum_{t=0}^{n} o_t$.

Note that, by taking this approach, our model will only be able to learn episodically, slowing down the learning process. Conversely, the belief approach to defining rewards has greater space complexity, and for large $\mathcal{S}$ will also have high time-complexity. Choosing which reward methodology to choose from is, really, more of an art than a science. The literature varies greatly, and you should intuit as follows: If observations fully inform rewards, take $R : \mathcal{O} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$, otherwise, if $\mathcal{S}$ is reasonably small in comparison to N, take $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, else take $R : \mathcal{T} \times \mathbb{R}$. There is a bit more nuance to this, e.g. if $N = \infty$ but our game usually terminates in just a few moves episodic reward will likely outperform belief-level reward, but in general this is a good guiding philosophy for the applied mathematician. We note that theorists, traditionally, take either the belief-level signal or episodic approach, whereas practitioners tend to use the observation signal or episodic approach, depending on the problem.

We summarise the three reward schemas covered in figure 1.

All together, we've described our imperfect information learning agent with the 9-tuple

$$(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbb{T}, \mathbb{T}_0, \mathbb{O}, R, \gamma, N)$$

Note, we justified adding $\gamma \in [0, 1)$ last section. The idea here is the same, though it is not necessary if we use episodic rewards $R : \mathcal{T} \to \mathbb{R}$. This tuple defines a *discrete partially observable Markov decision process.*

[12] For those familiar with recursive Bayesian estimation and/or the Kalman filter, that's how we usually update our beliefs.

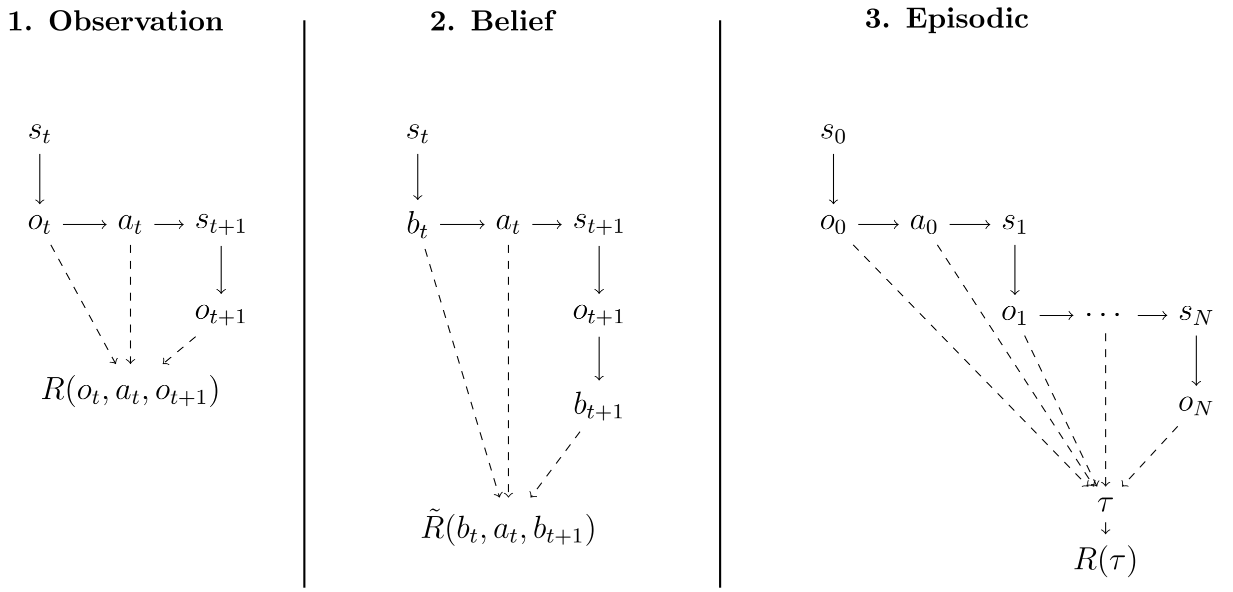

Figure 1: POMDP reward schemas

Definition 2. *(discrete POMDP) A discrete partially observable Markov decision process is a 9-tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbb{T}, \mathbb{T}_0, \mathbb{O}, R, \gamma, N)$ *where*

- $\mathcal{S}, \mathcal{A}, \mathcal{O}$ *are sets, named the state/action/observation spaces respectively.*
- $\mathbb{T}(s'|a, s) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *is a conditional distribution, named the transition distribution, describing the dynamics of state-action updates.*
- $\mathbb{T}(s) : \mathcal{S} \to [0, 1]$ *is a distribution, named initial state distribution, describing the dynamics of initial state selection.*
- $\mathbb{O}(o|s) : \mathcal{O} \times \mathcal{S} \to [0, 1]$ *is a conditional distribution, named the observation distribution, describing the dynamics of obtaining observations from hidden world states.*
- R *is a reward map, which can either be* $\mathcal{O} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$, $\mathbb{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *or* $(\mathcal{O} \times \mathcal{A})^N \times \mathcal{O} \to \mathbb{R}$ *depending on whether (A) observations are fully informative about rewards (B) whether beliefs* $b_t : \Delta(\mathcal{S}) \to [0, 1]$ *are held.*
- $\gamma \in [0, 1)$ *is the discount rate*
- N *is the time-horizon*

It remains to define the learning objective in this setting. Let $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbb{T}, \mathbb{T}_0, \mathbb{O}, R, \gamma, N)$ be a discrete POMDP. Earlier, the Markov assumption (assumption 1) allowed us to express a policy as a conditional distribution $\pi(a|s) : \mathcal{A} \times \mathcal{S} \to [0, 1]$. Naively, we'd expect that we can instead define our policies by instead conditioning on $o \in \mathcal{O}$. In practice this is *forgetful*, causing us to make decisions on incomplete information, due to the fact that $o \in \mathcal{O}$ does not contain all the information about the corresponding world state $s \in \mathcal{S}$. Recall the belief framework. As we obtain more observations, we expect to be more certain of our beliefs about the current state, despite this state changing stochastically[13].

We'll need to define N subpolicies, π_t, for each of the N levels of information depth we can have, to define a general global policy π. Each subpolicy π_t will take the up to time t, and use this to construct an action distribution. In particular, we will condition on the history up to time t, rather than just $s \in \mathcal{S}$. Let $\mathcal{T}_t = (\mathcal{O} \times \mathcal{A})^t \times \mathcal{O}$ be the space of *t-trajectoreis*, and let $\Delta(\mathcal{A})$ be the $\mathcal{A}$-simplex. Then, a *policy* for a POMDP is a family of maps $\pi = \{\pi_t : \mathcal{T} \to \Delta(\mathcal{A})\}_{t=1}^N$.

The learning objective is then defined analogously. We leave the details for each reward function case to the detail.

3 Exercises

1. Consider the game of chess, as laid out in example 1.
   (a) Describe the action space of the full game of chess, with castling, *en-passant*, and full pawn functionality.
   (b) Can we *prune* $\mathcal{S}, \mathcal{A}$ without losing any information? If so, how?
2. Let $(\mathcal{S}, \mathcal{A}, \mathbb{T}, \mathbb{T}_0, R, \gamma, N)$ be a Markov decision process. Let $\mathbb{T}(\cdot\,; \pi)$ denote the distribution of episodes τ sampled with policy π. Prove that, for all episodes τ,

$$\mathbb{T}(\tau; \pi) = \mathbb{T}_0(s_0) \prod_{t=1}^{N} \pi(a_t|s_t)\mathbb{T}(s_t|s_{t-1}, a_{t-1})$$

3. Fill in the details of proposition 1.
4. (harder) Derive the *episode distribution* for an POMDP with policy π.

[13] Or, at least, as certain as before, so certainty is weakly increasing.

References

Richard Bellman. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

Ronald A. Howard. Dynamic programming and markov processes. *The MIT Press*, 1960.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

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.