Markov Decision Processes Lecture Notes 01

Specification

Authors: Benjamin Van Roy *September 22, 2025*

1 MDP Specification via Transition Probabilities

A Markov decision process (MDP) models the evolution of an environment's state over time as it is influenced by actions applied by an agent and random events. The agent and environment interact through an interface as illustrated in Figure 1. At each time t, the agent observes a state S_t and applies an action A_t.

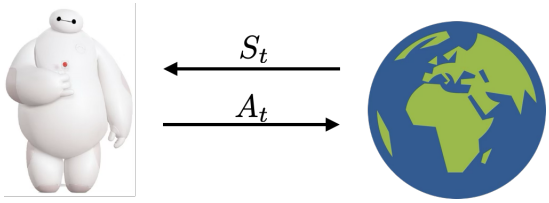

Figure 1: An agent-environment interface.

1.1 The Tuple

When the state space is finite or countably infinite, it is common to specify an MDP via a tuple $(\mathcal{S}, \mathcal{A}, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and P encodes transition probabilities. In particular, for any states $s, s' \in \mathcal{S}$ and action $a \in \mathcal{A}$, $P_{ass'}$ is the probability that the next state will be s' if the current state is s and the action a is applied. In other words, each state realization S_{t+1} is sampled according to probabilities $(P_{A_t,S_t,s} : s \in \mathcal{S})$.

1.2 Examples

1.2.1 Queueing

Consider a queue of customers waiting at a service station, as illustrated in Figure 2. At each time, an additional customer arrives with probability q. The station can operate in a fast or slow mode of service. If there are customers in the queue, the fast mode removes one with probability p_{fast} and the slow mode, p_{slow}. The fast mode is more expensive and only used as wait times become large.

Figure 2: Queueing at a service station.

It is natural to model queue length dynamics in terms of an MDP $(\mathcal{S}, \mathcal{A}, P)$ with $\mathcal{S} = \{0, 1, 2, \ldots\}$ and $\mathcal{A} = \{\text{fast}, \text{slow}\}$. Each state S_t is the current queue length, and the action A_t indicates whether the fast or slow service mode is applied. Transition probabilities are provided in Table 1.

$P_{ass'}$	$s' = s - 1$	$s' = s$	$s' = s + 1$
$s = 0$	0	$1 - q$	q
$s > 0$	$(1 - q)p_a$	$qp_a + (1 - q)(1 - p_a)$	$q(1 - p_a)$

Table 1: Queue transition probabilities. The rows provide formulas that apply when $s = 0$ or $s > 0$, respectively.

1.2.2 Routing

Consider a directed graph with vertices $\mathcal{V}$ and directed edges $\mathcal{E}$. Navigation through the graph can be modeled in terms of an MDP $(\mathcal{S}, \mathcal{A}, P)$, with $\mathcal{S} = \mathcal{V}$, $\mathcal{A} = \mathcal{V}$, and

$$P_{ass'} = \begin{cases} 1 & \text{if } (s, a) \in \mathcal{E} \text{ and } s' = a \\ 0 & \text{otherwise.} \end{cases}$$

The state S_t indicates a current vertex, and the action A_t identifies a next vertex to target. The state S_{t+1} is A_t if there is an edge but otherwise remains at $S_{t+1} = S_t$. Actions route the state, forming a path through the graph.

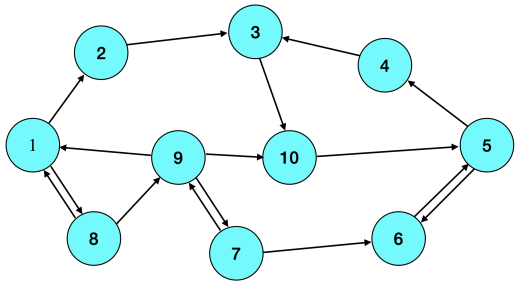

Figure 3: A directed graph.

Figure 3 provides an example where $\mathcal{V} = \{1, \ldots, 10\}$. If $S_0 = 1$ then an action $A_0 = 8$ transitions to $S_1 = 8$. On the other hand, an action $A_0 = 9$ would result in $S_1 = 1$ because there is no edge leading from vertex 1 to vertex 9.

1.2.3 Tetris

The game of Tetris is naturally modeled as an MDP $(\mathcal{S}, \mathcal{A}, P)$. The state encodes the current configuration of the 20x10 cell Tetris board, as illustrated in Figure 4. The action set is $\mathcal{A} = \{\text{left}, \text{right}, \text{drop}, \text{none}\}$. Transition probabilities $P_{ass'}$ encode dynamics. This includes how the falling tetromino moves as actions are applied, how rows of the brick wall vanish when complete, and how new tetromino's are sampled and appear.

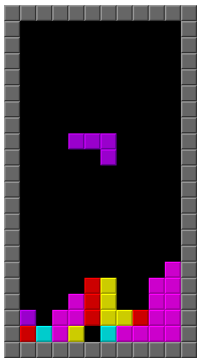

Figure 4: A Tetris board.

2 MDP Specification via Transition Function

There are multiple equivalent ways of specifying an MDP. In this section, we introduce an alternative to the $(\mathcal{S}, \mathcal{A}, P)$ tuple. This alternative facilitates a more intuitive interpretation of some models, for example, where the state space is a Euclidean space or most naturally thought of as embedded in a Euclidean space.

2.1 The Tuple

We consider specifying an MDP in terms of a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{W}, f, \nu)$, where $\mathcal{W}$ is the disturbance space, f is a transition function, and ν is a probability distribution over $\mathcal{W}$. States generated by the MDP then evolve according to $S_{t+1} = f(S_t, A_t, W_{t+1})$, where W_{t+1} is sampled independently from $\nu(\cdot|S_t, A_t)$.

Consider an MDP with a countable state and action spaces. By *countable*, we mean finite or countably infinite. Without loss of generality we can index states with natural numbers so that $\mathcal{S} = \{1, 2, 3, \ldots, N\}$, where N could be a positive integer or infinity. Such an MDP can alternately be specified by $(\mathcal{S}, \mathcal{A}, \mathcal{W}, f, \nu)$ or $(\mathcal{S}, \mathcal{A}, P)$. Given the former specification, the latter can be derived by letting $P_{ass'} = \nu(\{w \in \mathcal{W} : s' = f(s, a, w)\})$. Given the latter specification, the former can be derived by letting each $\nu(\cdot|s, a)$ be a uniform distribution over the unit interval and $f(s, a, w) = s'$ if $w \in [p_{a,s,s'-1}, p_{ass'})$, where $p_{ass'} = \sum_{s''=1}^{s'} P_{ass''}$ for $s' = 0, 1, 2, \ldots, N$.

2.1.1 Examples

2.1.2 Linear Systems

A stochastic linear system is naturally modeled as an MDP $(\mathcal{X}, \mathcal{U}, \mathcal{W}, f, \nu)$. Note that we have denoted the state and action spaces by $\mathcal{X}$ and $\mathcal{U}$ to be consistent with the linear systems literature. The state and disturbance spaces are $\mathcal{X} = \mathcal{W} = \mathbb{R}^N$ and the action space is $\mathcal{U} = \mathbb{R}^M$. The update function is

$$f(x, u, w) = Ax + Bu + w.$$

The disturbance distribution is often assumed to be Guassian: $\nu(\cdot|s, a) \sim \mathcal{N}(0, \Sigma)$.

To offer a concrete instance of such a stochastic linear system, consider pursuit of a target in one dimension, along the lines illustrated in Figure 5. Let $x_{t,1}$ and $x_{t,2}$ be the position and velocity along that dimension of a vehicle. Let u_t be the force applied to accelerate. These state variables then evolve according to

$$x_{t+1,1} = x_{t,1} + x_{t,2} \qquad \text{and} \qquad x_{t+1,2} = x_{t,2} + \frac{u_t}{m},$$

where m denotes the mass of the vehicle. The expression u_t/m derives from Newton's second law: force = mass $\times$ acceleration. Let $x_{t,3}$ be the position of a target, which moves randomly according to

$$x_{t+1,3} = x_{t,3} + \zeta_{t+1},$$

where ζ_{t+1} is iid $\mathcal{N}(0, \sigma^2)$. This system is naturally modeled as an MDP $(\mathcal{X}, \mathcal{U}, \mathcal{W}, f, \nu)$ with $\mathcal{X} = \mathbb{R}^3$, $\mathcal{U} = \mathbb{R}$, $\mathcal{W} = \mathbb{R}^3$, $f(x, u, w) = Ax + Bu + w$, where

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1/m \\ 0 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}.$$

Figure 5: Pursuit of a target in one dimension.

We could alternatively formulate an MDP with a two-dimensional state space, focusing on the difference in position between the vehicle and the target rather than the two positions individually. In particular, let $x_{t,1}$ be the difference in position, and let $x_{t,2}$ be the velocity. The dynamics of these quantities can be modeled by an MDP $(\mathcal{X}, \mathcal{U}, \mathcal{W}, f, \nu)$ with $\mathcal{X} = \mathbb{R}^2$, $\mathcal{U} = \mathbb{R}$, $\mathcal{W} = \mathbb{R}^2$, $f(x, u, w) = Ax + Bu + w$, where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Whether this second MDP is a suitable alternative for our purposes depends on whether our objective depends only on the distance between the vehicle and target or also on the individual positions of the two objects. As we will see in a future lecture, we accommodate objectives obtained by combining rewards, each determined by a state-action pair. States must encode information required to compute rewards.

2.1.3 Inventory

Consider a retailer that sells one product. At each time, the retailer holds some number of units in stock and can place an order for more. Within each timestep, a random number of customers make purchases. Ordered units arrive at the end of the timestep.

Such an inventory system can be modeled as an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{W}, f, \nu)$, with state, action, and disturbance spaces $\mathcal{S} = \mathcal{A} = \mathcal{W} = \{0, 1, 2, \ldots\}$. The state is the current inventory level, the action is the number of units ordered, and the disturbance is the demand. The update function is

$$f(s, a, w) = \max(s + a - w, 0).$$

Hence, the next inventory level S_{t+1} is the sum of the current level S_t and the order quantity a minus the demand w, unless this is negative, in which case the inventory level becomes zero. The demand distribution could be $\nu(\cdot|s, a) \sim \text{Poisson}(\lambda)$ to model the number of customers arriving within a timestep if they arrive at rate λ according to a Poisson process.

We could alternatively model an inventory system that accommodates backorders. This can be done with a state space $\mathcal{S} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Negative states indicate backorders: units of demand from customers that arrived over previous timesteps and who are waiting for their purchases to be filled. For such an inventory system, the update function is

$$f(x, u, w) = s + a - w.$$

2.1.4 Investment

Consider an investor who at each time balances his wealth between one risk-free and N risky securities. The dynamics of his wealth can be modeled via an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{W}, f, \nu)$, with $\mathcal{S} = \mathbb{R}_+$, $\mathcal{A} = \{a \in \mathbb{R}_+^N : a \geq 0, \sum_{n=1}^N a_n \leq 1\}$, and $\mathcal{W} = \mathbb{R}_+^{N+1}$. The state is his current wealth in dollars, the action indicates investments in each of the N risky securities as fractions of wealth, and the disturbance provides returns for the risky securities. The return of the riskless security is a fixed constant z. The update function is

$$f(s, a, w) = zs + \sum_{n=1}^{N} (w_n - z) a_n s.$$

Note that w_0 is the return of the riskless security, and $w_1, \ldots, w_N$ are the returns of the risky securities.

States thus evolve according to

$$S_{t+1} = f(S_t, A_t, W_{t+1}).$$

Each riskless return $W_{t+1,0}$ is deterministic. Risky returns could be modeled as $W_{t+1,n} = e^{X_{t+1,n}}$, where $X_{t+1} \sim \mathcal{N}(0, \Sigma)$ is an independent N-dimensional random Gaussian vector. This induces a multidimensional lognormal distribution $\nu(\cdot|s, a)$. Note that it is more natural to model returns as lognormal rather than normal because they are always positive.

2.1.5 Dialog

Consider an agent that engages in sequential correspondence, with information conveyed via messages exchanged with another party. The agent assigns an initial state S_0, which indicates that the conversation has not yet begun. The agent then transmits a message A_0 and receives a response O_1. Such exchanges continue. This interaction could be mediated, for example, by a text interface. For example, the first message A_0 could be "How can I help you?" The response O_1 could then be "How does one replace a light bulb?" Subsequent messages transmitted by the agent could seek clarification regarding the task at hand and guide the process through completion.

To study the dynamics of such dialogue, we could assume that responses are generated by a language model. Figure 6 provides a block diagram that illustrates the operation of a recurrent neural network (RNN) language model. The RNN streams tokenized text. Tokenization breaks down text into small units called tokens. Each token typically represents a word, some form of punctuation, or an end-of-message indicator. Let K denote the size of the token dictionary. The RNN maintains a state vector, which is updated after each token. The RNN generates a predictive distribution $p(\cdot|s)$ of the next token by multiplying this state $s \in \mathbb{R}^N$ by an $K \times N$ unembedding matrix B and then applying a softmax. In other words, the probability assigned to token k is $p(\cdot|s) = e^{(Bs)_k} / \sum_{k'=1}^K e^{(Bs)_{k'}}$.

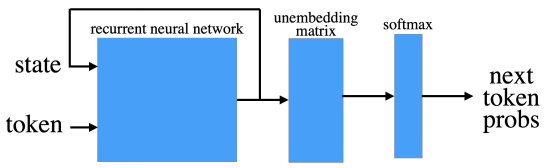

Figure 6: A recurrent neural network language model.

To model dialog in terms of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{W}, f, \nu)$, we take take the state space to be the RNN state space $\mathcal{S} = \mathbb{R}^N$ and the action and disturbance spaces to be the set of possible messages $\mathcal{A} = \mathcal{W} = \{x \cup K : x \in \cup_{m=1}^N \{1, \ldots, K-1\}^m\}$, where the end-of-message indicator is indexed by K. The disturbance W_{t+1} is the next response. Its distribution $\nu(\cdot|S_t, A_t)$ is induced by the RNN. In particular, a response consisting of tokens $\tau_1, \ldots, \tau_M = K$ is assigned probability $\prod_{m=1}^M p(\tau_m|s_m)$, where s_0 is the state immediately preceding

this response and $s_{m+1} = s_m \cup \tau_m$. Finally, $f(S_t, A_t, W_{t+1})$ is the state of the RNN after processing the entire dialogue through the end of the response W_{t+1}.

3 Policies

The history $H_t = (S_0, A_0, S_1, A_1, \ldots, S_t)$ records past states and actions. Let $\mathcal{H} = \mathcal{S} \times \cup_{t=0}^{\infty} (\mathcal{A} \times \mathcal{S})^t$ be the set of all such finite sequences. A policy π maps the history to action probabilities. In particular, for each $h \in \mathcal{H}$ and $a \in \mathcal{A}$, $\pi(a|h)$ denotes the probability assigned to action a given history h. Under a policy π, each action A_t is sampled from $\pi(\cdot|H_t)$.

3.1 Stationary Policies

A policy is *stationary* if it depends on the history H_t only through the most recent state S_t. With some abuse of notation, for a stationary policy π, we alternatively write action probabilities as $\pi(\cdot|S_t) \equiv \pi(\cdot|H_t)$.

3.2 Deterministic Stationary Policies

A deterministic stationary policy is a stationary policy that, for each state, assigns all probability to a single action. In other words, for state each $s \in \mathcal{S}$, there is an action $a \in \mathcal{A}$ such that $\pi(a|s) = 1$. With some abuse of notation, for a deterministic stationary policy, we sometimes write that action a as $\pi(s)$.

3.3 Examples

To offer an example, for the queueing system of Section 1.2.1, it is natural to consider a policy that applies the fast service mode when the queue length exceeds some threshold and, otherwise, the slow mode. This is a deterministic stationary policy. If the threshold is ten, this policy takes the form

$$\pi(s) = \begin{cases} \text{fast} & \text{if } s > 10 \\ \text{slow} & \text{otherwise.} \end{cases}$$

It might also be natural to consider a stochastic (not deterministic) stationary policy that applies the fast model with a probability that increases with queue length. For example, we could have

$$\pi(\text{fast}|s) = 1 - e^{s/10} \qquad \text{and} \qquad \pi(\text{slow}|s) = e^{s/10}.$$

The so-called $(\overline{S}, \overline{s})$ *policy* is commonly applied to inventory systems of the sort discussed in Section 2.1.3. This policy is parameterized by two thresholds, often denoted by $\overline{S}$ and $\overline{s}$. The idea is to order the number of units required to bring the inventory level up to $\overline{S}$ whenever the level is at or below $\overline{s}$. If $\overline{S} = 100$ and $\overline{s} = 50$, this deterministic stationary policy takes the form

$$\pi(s) = \begin{cases} \overline{S} - s & \text{if } s \leq \overline{s} \\ 0 & \text{otherwise.} \end{cases}$$

4 Dynamics, Objectives, and Optimization Algorithms

An MDP represents a hypothesis about how the state of an environment evolves over time and how that evolution can be influenced by an agent's actions. Given an MDP and a policy, one can analyze **dynamics**; that is, how the environment behaves under this hypothesis and policy. For example, one could predict the fraction of time the environment will spend in a particular state and the average and standard deviation of sojourn times.

To offer prescriptive advice on what policy to use, we need an **objective**. We will formulate objectives in terms of reward functions $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that attribute rewards $R_{t+1} = r(S_t, A_t)$ to state-action pairs. Hence,

the reward function generates a scalar sequence $R_1, R_2, R_3, \ldots$. To arrive at a single scalar objective, we must combine these rewards in some way. For example, we could compute the *discounted return* $\sum_{t=0}^{\infty} \gamma^t R_{t+1}$, where $\gamma \in [0, 1)$ is a discount factor that assigns a level of priority to nearer over longer term rewards. The discounted return depends on the random future evolution of state. An ex-ante analysis must instead be based on an expectation $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right]$, which averages over random future outcomes.

Given an MDP, reward function, and way of combining rewards accrued over times, there are a variety of **optimization algorithms**. Such algorithms aim to identify optimal, near-optimal, or, at least, satisficing policies.

Over the next few lectures, we will cover approaches to analyzing dynamics, specifying objectives, and designing and applying optimization algorithms.