

# Markov Decision Processes and Reinforcement Learning

## Day 1 — Definition of MDP and Examples

Luiz Felipe Martins

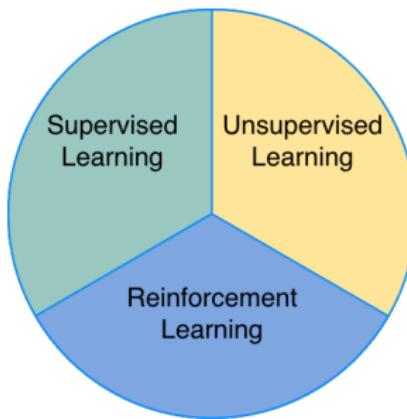
Department of Mathematics and Statistics  
Cleveland State University

January 27, 2026

# Outline

- 1 Introduction and Goals
- 2 Markov Chains
- 3 Markov Decision Processes — Informal Description
- 4 Markov Decision Processes — Mathematical Formulation

# The Three ML Paradigms



- **Supervised Learning:** Model is given input–output pairs and learns to predict the correct output for new, unseen inputs.
- **Unsupervised Learning:** Model discovers structure or patterns without predefined target values.
- **Reinforcement Learning:** A model for sequential decision making, where the agent learns how to perform optimally in an environment.

# What is Reinforcement Learning?

- In a Reinforcement Learning (RL) model, an agent acts on an environment.
- Depending on an action chosen by the agent, the state of the environment changes and the agent receives a numerical reward.
- The agent's goal is to choose actions in a way that long-term rewards are maximized.
- Historically, RL was independently developed in different areas. It is also known as Dynamic Programming, Markov Decision Processes (MDPs) and Sequential Decision Models.

# Areas of Application

Application Area	Concrete Examples
Robotics & Control	Robotic grasping and assembly; drone navigation; quadruped locomotion; industrial pick-and-place systems.
Games & Decision Making	AlphaGo/AlphaZero (Go, Chess); Atari agents (DQN); StarCraft II and Dota 2 self-play systems.
Recommendation & Personalization	YouTube video ranking; e-commerce product ordering; personalized news feeds; adaptive educational tutors.
Autonomous Systems	Self-driving vehicles; warehouse AGVs; traffic light coordination; spacecraft docking.
Operations & Resource Allocation	Cloud job scheduling; dynamic pricing; energy grid control; supply chain routing and inventory management.
Finance & Trading	Portfolio optimization; algorithmic trading; market making; sequential fraud detection.
Healthcare & Medicine	Optimal drug dosing; radiotherapy planning; clinical decision support; robotic rehabilitation systems.

# Workshop Goals

Goals:

- Understand the mathematical framework used by RL models.
- Learn about both classical and modern solution methods.
- Understand how to set up an RL model in an applied problem.
- Learn how to translate a mathematical RL model into a computational model.
- Introduce current libraries for solving RL problems.

# Intuitive Description of a Markov Chain

- A **Markov Chain** is a probabilistic model for a process that evolves in time. It is a particular kind of discrete-time stochastic process.
- A key feature of a Markov Chain is that, at any time, the future evolution of the process depends only on the current state, and not on previous states visited by the chain.
- This seems to be restrictive, but it is a common assumption in mathematical modeling. It can usually be achieved by enlarging the state space to include all information needed to predict the future of the process.
- My favorite example: Isaac Newton realized that, to completely predict the motion of a particle in a force field, it is necessary to know both its initial *position* and *velocity*.

## Definition of Markov Chain

- Let  $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$  be a finite set. We call this set the **state space** of the chain, or **environment**.
- A **Markov Chain** on  $\mathcal{S}$  is a stochastic sequence  $\{S_t\}_{t \in \mathbb{N}}$  on  $\mathcal{S}$  such that:

$$\mathbb{P}(S_{t+1} = s_j | S_t = s_i, S_{t-1}, S_{t-2}, \dots, S_0) = \mathbb{P}(S_t = s_j | S_{t-1} = s_i)$$

for all  $t \geq 0$ . This is called the **Markov Property**.

- The values:

$$\mathbb{P}(S_t = s_j | S_{t-1} = s_i)$$

are called **transition probabilities**.

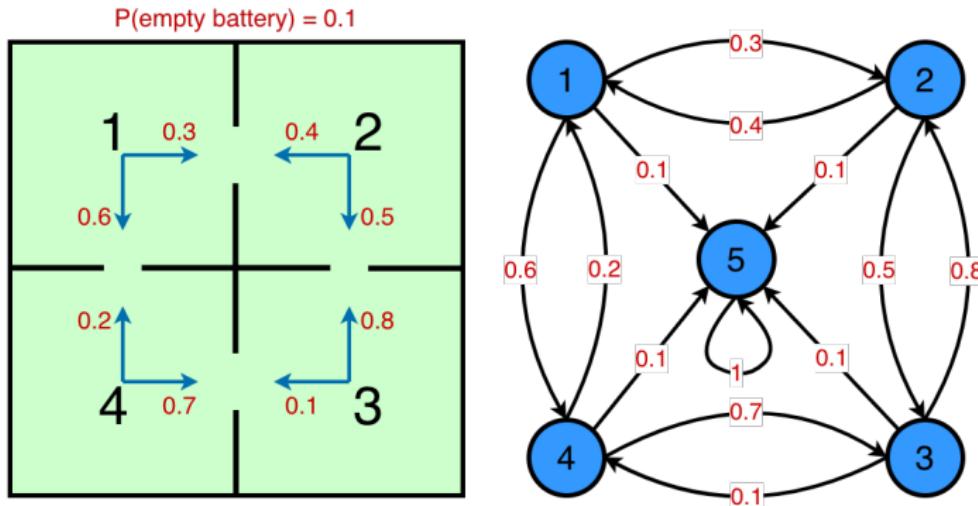
# Transition Probability Matrix

- We assume that our Markov Chain is **time-invariant**, meaning that the transition probabilities are independent of  $t$ .
- The **transition probability matrix**  $P$  of a Markov Chain  $\{S_t\}_{t \in \mathbb{N}}$  is defined by:

$$P_{ij} = \mathbb{P}(S_{t+1} = s_j \mid S_t = s_i)$$

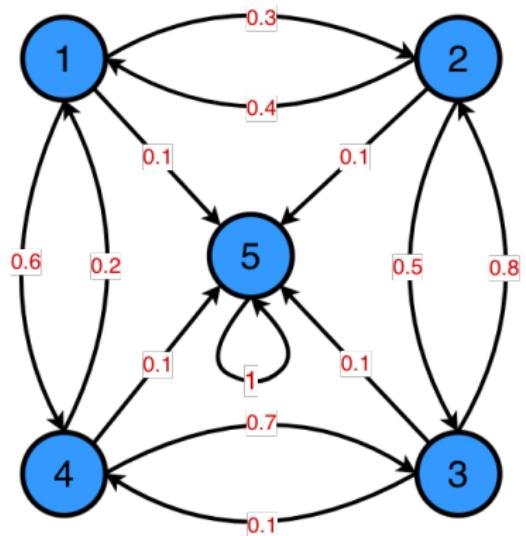
- Everything we need to compute for a Markov Chain can be expressed in terms of the matrix  $P$ .

# Markov Chain Example



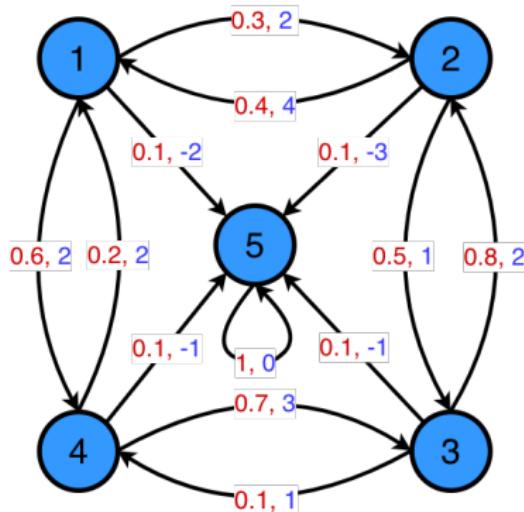
- Robot operates in a 4-room environment, in each step it transitions to another room with probabilities as given in the figure.
- Each step there is a probability of 0.1 that the robot's battery runs out, and the process terminates.

# Transition Probability Matrix



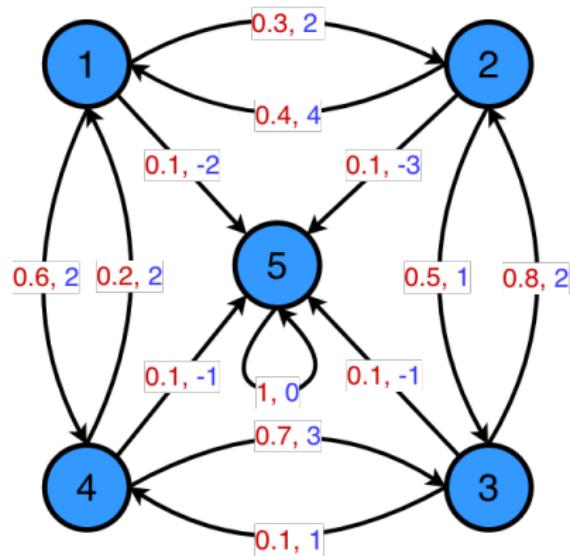
$$P = \begin{bmatrix} 0 & 0.3 & 0 & 0.6 & 0.1 \\ 0.4 & 0 & 0.5 & 0 & 0.1 \\ 0 & 0.8 & 0 & 0.1 & 0.1 \\ 0.2 & 0 & 0.7 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

# Running Rewards



- A transition from  $s$  to  $s'$  yields a **reward**  $r(s, s')$ .
- A negative reward represents a penalty.
- Once the battery empties, the process stops.

# Rewards Matrix



$$R = \begin{bmatrix} - & 2 & - & 2 & -2 \\ 4 & - & 1 & - & -3 \\ - & 2 & - & 1 & -1 \\ 2 & - & 1 & - & -1 \\ - & - & - & - & 0 \end{bmatrix}$$

- If the probability of a transition from  $s$  to  $s'$  is zero, the corresponding entry in the rewards matrix is irrelevant, and is marked “—”.
- Usually, these are just set to 0.

# Markov Chain Example — Expected Total Reward

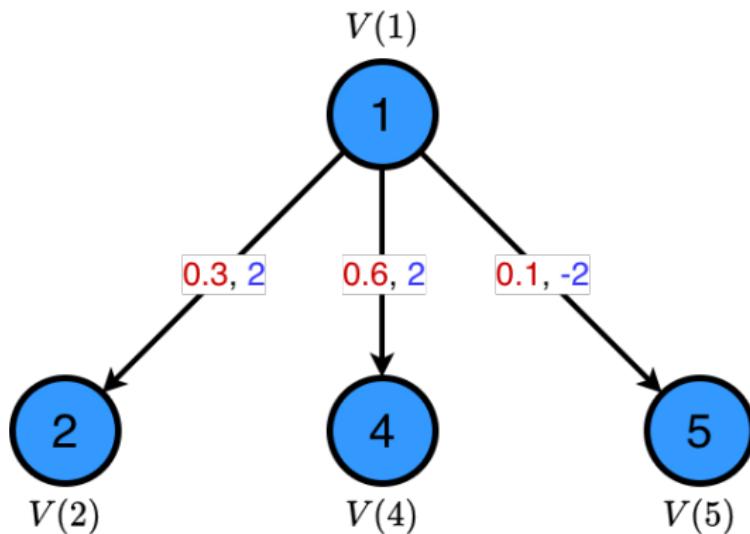
- Let  $R_{t+1} = r(S_t, S_{t+1})$ .
- The **expected total reward** is defined by:

$$\mathbb{E} \left[ \sum_{t=0}^T R_{t+1} \right]$$

- $T$  represents the termination time. Notice that  $P(T < \infty) = 1$ .
- The **state value function** is defined as the expected total reward:

$$V(s) = \mathbb{E} \left[ \sum_{t=0}^T R_{t+1} \mid S_0 = s \right]$$

# One-step Analysis



$$V(1) = 0.3(2 + V(2)) + 0.6(2 + V(4)) + 0.1(-2 + V(5))$$

# Computation of the Value Function

$V$  is the solution of the *linear system*:

$$V(1) = 0.3(2 + V(2)) + 0.6(2 + V(4)) + 0.1(-2 + V(5))$$

$$V(2) = 0.4(4 + V(1)) + 0.5(1 + V(3)) + 0.1(-3 + V(5))$$

$$V(3) = 0.8(2 + V(2)) + 0.1(1 + V(4)) + 0.1(-1 + V(5))$$

$$V(4) = 0.2(2 + V(1)) + 0.7(1 + V(3)) + 0.1(-1 + V(5))$$

$$V(5) = 0$$

Solution:

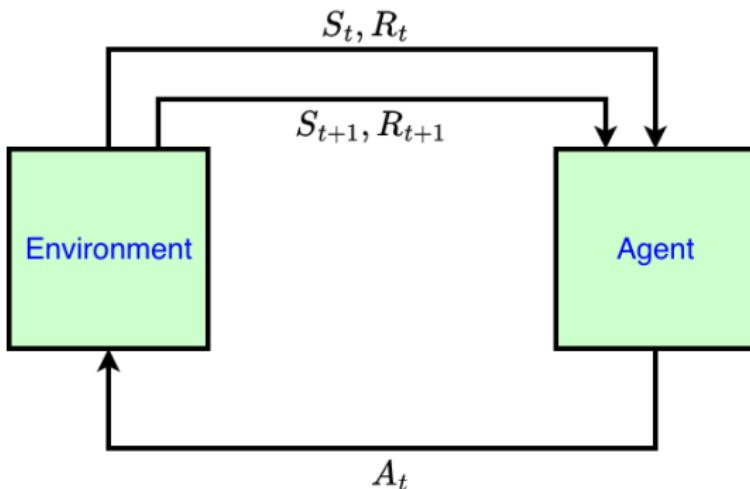
$$V(1) = 15.512 \quad V(2) = 15.943 \quad V(3) = 15.876$$

$$V(4) = 15.215 \quad V(5) = 0$$

# The Components of an RL Model

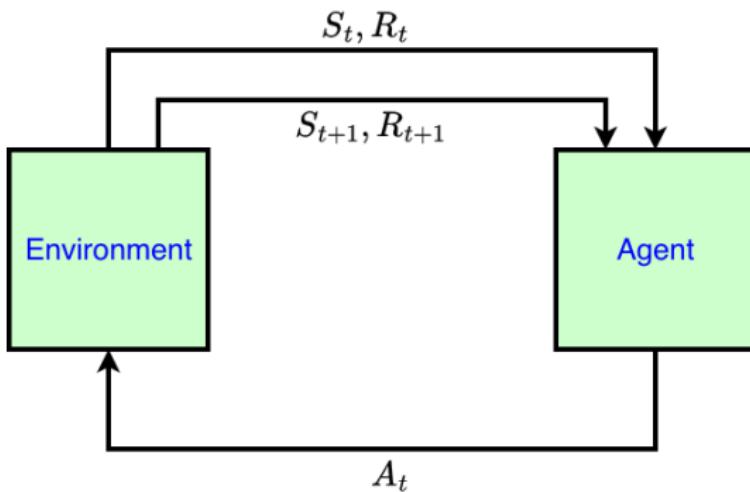
- The **environment** represents all information that is relevant for the optimization problem. It keeps a record of its *current state*, and provide the agent with information about *observations* and *rewards*.
- The **agent** interacts with the environment, obtaining *observations* (partial representations of the environment) and *rewards*. The rewards can be negative, representing penalties.
- The agent chooses **actions**. The actions determine both the probabilistic evolution of the environment and the rewards obtained by the agent.
- The agent's goal is to **maximize expected rewards** on a run of interactions with the environment.

# The Agent-Environment Interaction



- The **observes** the environment, and chooses an **action**
- As a result of the action, the environment changes **state**.
- The agent receives a **reward**, dependent on the state of the environment and the action chosen.

# A simplification



- We assume that, when choosing an action, the agent has *complete information about the state of the environment*.
- In the general case, the observation can be a subset of the state.

# Definition of Markov Decision Process

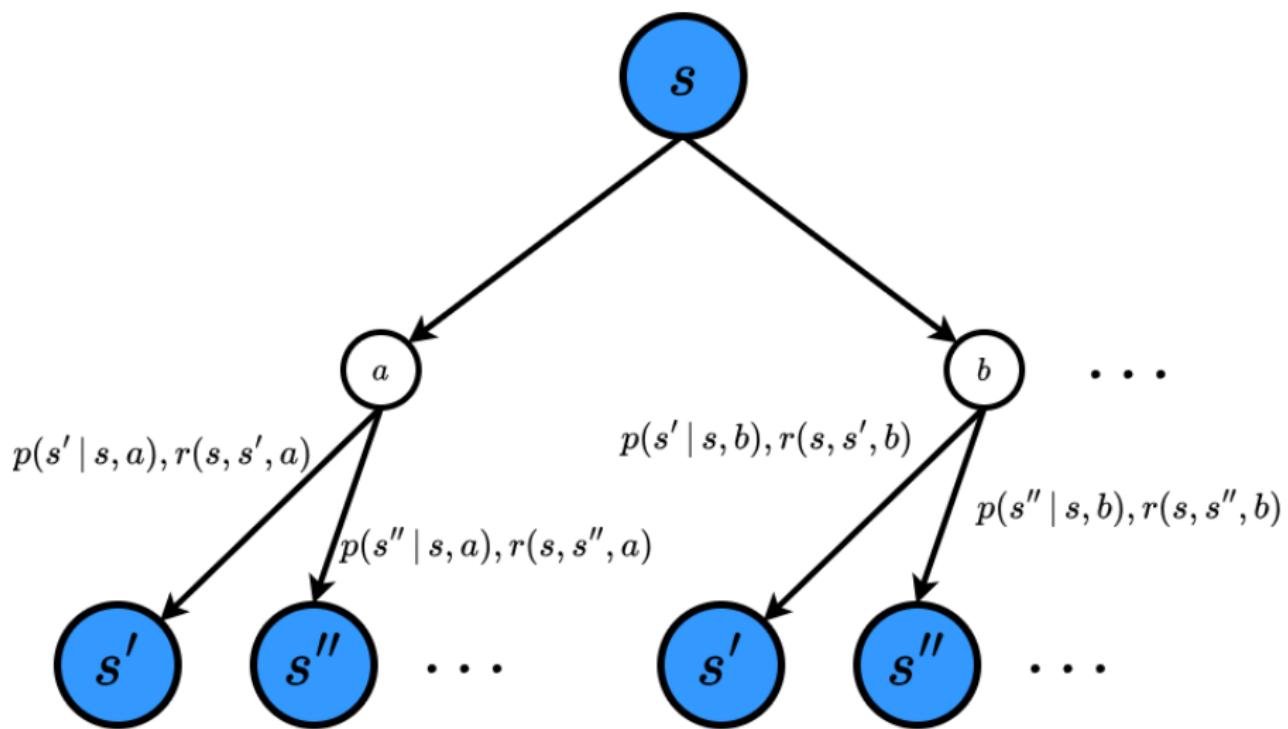
A **Markov Decision Process** (MDP) consists of:

- A finite set  $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$ , the set of states the environment can be in.
- For each  $s \in \mathcal{S}$ , a finite set  $\mathcal{A}(s)$ , the set actions available to the agent when the environment is in state  $s$ .
- Two functions of three variables:

$p(s' | s, a)$  representing *transition probabilities*

$r(s, s', a)$  representing *running rewards*

# Graphical Representation



# Transition probabilities

The **transition probability matrix**  $P^{(a)}$  associated to action  $a$  is defined as follows:

$$P_{ij}^{(a)} = p(s_j | s_i, a)$$

We require:

$$0 \leq P_{ij}^{(a)} \leq 1, \quad \sum_{j=1}^N P_{ij}^{(a)} = 1$$

**Note:**  $P_{ij}^{(a)}$  is defined arbitrarily if action  $a$  is not available for state  $s_i$ .

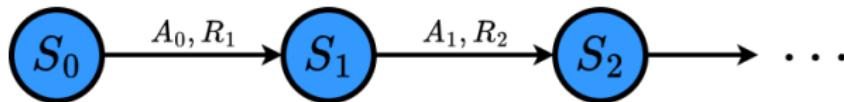
# Running Rewards

- We assume that the reward received for a transition from  $s$  to  $s'$  under action  $a$  is given by a *deterministic function*:

$$r(s, s', a)$$

- Computational frameworks allow for randomized rewards.
- The **reward matrix**  $R^{(a)}$  associated with action  $a$  is defined as:

$$R_{ij}^{(a)} = r(s_i, s_j, a)$$



A **trajectory** of a MDP is represented by three sequences of random variables:

- $S_0, S_1, S_2, \dots$ : the successive *states* of the environment.
- $A_0, A_1, A_2, \dots$ , the successive *actions* taken by the agent.
- $R_0, R_1, R_2, \dots$ , the successive *rewards* obtained by the agent.

These sequences are related as follows:

$$\mathbb{P}(S_{t+1} = s' | S_t = s) = p(s' | s, A_t)$$

$$R_{t+1} = r(S_t, S_{t+1}, A_t)$$

# Curse of Dimensionality

- The number of possible states is usually astronomically large. This phenomenon is known as the **curse of dimensionality**.
- Example: the game **2048**

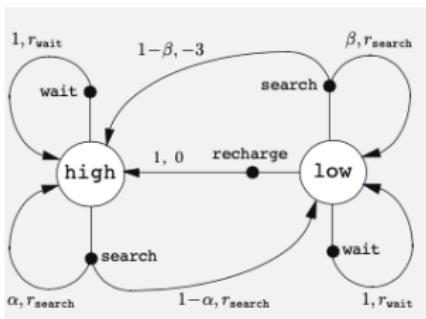


- Each cell can be empty or contain a power of two, between 2 and  $2048 = 2^{11}$ . Thus, the number of possible states the environment can be in is:

$$12^{16} = 184852952865954416.$$

- In practical applications, it is impossible to tabulate all values of transition probabilities.

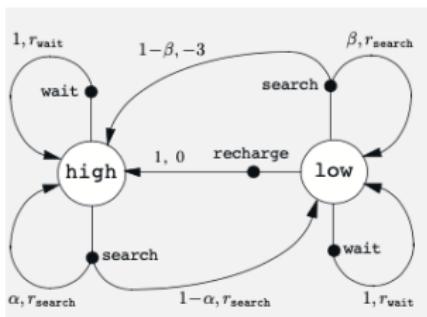
# Example 1: Recycling Robot



- $\mathcal{S} = \{\text{high}, \text{low}\}$  representing the battery levels of the robot.
- $\mathcal{A}(\text{high}) = \{\text{wait}, \text{search}\}$ . When in the high state, the robot can do nothing or search for a soda can.
- $\mathcal{A}(\text{low}) = \{\text{wait}, \text{search}, \text{recharge}\}$ . When in the low state, the robot, besides waiting and searching, can go to a recharging station.

(From R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*)

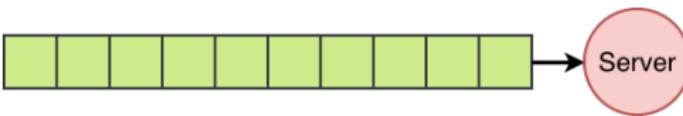
# Transition Probabilities and Reward Vectors



$$P^{\text{search}} = \begin{bmatrix} \alpha & 1 - \alpha \\ 1 - \beta & \beta \end{bmatrix} \quad P^{\text{wait}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad P^{\text{recharge}} = \begin{bmatrix} - & - \\ 1 & 0 \end{bmatrix}$$

$$R^{\text{search}} = \begin{bmatrix} r_{\text{search}} & r_{\text{search}} \\ -3 & r_{\text{search}} \end{bmatrix} \quad R^{\text{wait}} = \begin{bmatrix} r_{\text{wait}} & - \\ - & r_{\text{wait}} \end{bmatrix} \quad R^{\text{recharge}} = \begin{bmatrix} - & - \\ 0 & - \end{bmatrix}$$

## Example 2 — Server Optimization: System Evolution

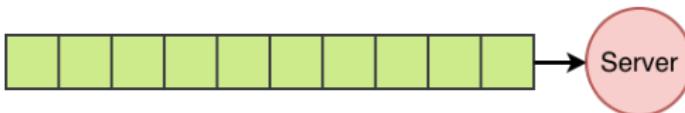


- A computer server receives service requests. Requests are queued in the order they arrive.
- Each second, there is a probability  $p$  that a new request arrives. The probability that two or more requests arrive in a second is negligible.
- The server has three modes of operation, high, normal and low. Every second, the probabilities that a request is completed are:

$$q_{\text{high}} > q_{\text{normal}} > q_{\text{low}} > 0$$

- The system can hold at most  $M$  requests. Requests that arrive when the system is full are rejected.

## Example 2 — Server Optimization: Costs



Each second, following running costs occur:

- For each service mode, there is an *operation cost*, where:

$$c_{\text{high}} > c_{\text{normal}} > c_{\text{low}} > 0$$

- If there are  $N$  items in the system (being served or waiting for service), there is a *holding cost*  $h(N)$ . Function  $h$  is a positive, increasing function.
- Whenever an item is rejected due to a full system, a *service loss cost*  $c_{\text{loss}}$  is incurred.

# Transition Probabilities

Each instant in time, if there are  $i$  items in the system, the only possible transitions are to  $i$ ,  $i + 1$  and  $i - 1$ . Let  $\text{mode} \in \{\text{high}, \text{normal}, \text{low}\}$ :

- If  $1 \leq i \leq M$ :

$$p(i+1 | i, \text{mode}) = p(1 - q_{\text{mode}})$$

$$p(i | i, \text{mode}) = pq_{\text{mode}} + (1 - p)(1 - q_{\text{mode}})$$

$$p(i-1 | i, \text{mode}) = (1 - p)q_{\text{mode}}$$

- For an empty system:

$$p(1 | 0, \text{mode}) = p, \quad p(0 | 0, \text{mode}) = 1 - p$$

- For a full system:

$$P(M-1 | M, \text{mode}) = (1 - p)q_{\text{mode}}, \quad P(M | \text{mode}) = 1 - (1 - p)q_{\text{mode}}$$

# Rewards

Suppose that there are  $i$  items in the system and mode of operation mode is selected.

- If  $0 \leq i \leq M$ :

$$r(i, \text{mode}) = -(h(i) + c_{\text{mode}})$$

- For a full system:

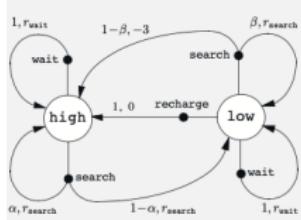
$$r(M, \text{mode}) = -(h(M) + c_{\text{mode}} + p(1 - q_{\text{mode}})c_{\text{loss}}).$$

# Episodic and Continuing Tasks

- In an **episodic task**, runs terminate after a finite number of steps with probability 1.



- In a **continuing task**, runs never terminate, and the agent operates in the environment forever.



# Terminal States

- In an episodic task, a **terminal state** is a state  $s$  such that:

$$p(s | s, a) = 1 \text{ and } r(s, s, a) = 0$$

for every action  $a$ .

- For any sequence of actions, a terminal state is eventually reached with probability one.
- We let  $T$  represent the (random) time a terminal state is first reached.

# Total Return for Episodic Tasks

- The **total return** accumulated in a run of an episodic task is:

$$\sum_{t=0}^T R_{t+1}$$

- This does not work for continuing tasks, since the sum would be infinite!

# Total Return for Continuing Tasks

- The total reward for a continuing task is defined as:

$$\sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

- The number  $\gamma$  is called **discount factor**, and we assume  $0 < \gamma < 1$ . This definition is motivated by the concept of *net present value* from economics.
- Under assumption of finiteness of states and actions, rewards are uniformly bounded, so the series always converges.

# Unified Notation for Episodic and Continuing Tasks

We will use, both for episodic and continuing tasks, the following notation for total returns:

$$\sum_{t=0}^T \gamma^t R_{t+1}$$

- For *episodic tasks*, we require  $P(T < \infty) = 1$  and  $0 < \gamma \leq 1$ .
- For *continuing tasks*, we assume  $P(T = \infty) = 1$  and  $0 < \gamma < 1$ .

# Deterministic policies

- A **deterministic policy** is a function  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  such that  $\pi(s) \in \mathcal{A}(s)$  for all  $s$ .
- Once a deterministic policy is chosen, the evolution of the stochastic sequence  $(S_t, A_t, R_t)$  is completely determined:

$$A_t = \pi(S_t)$$

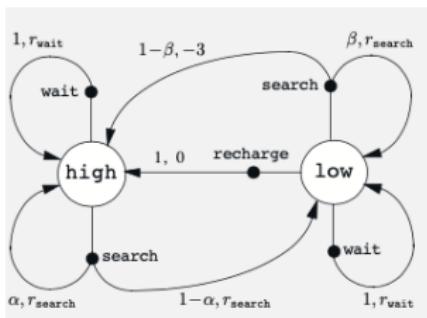
$$P(S_{t+1} = s' | S_t = s) = p(s' | S_t, A_t)$$

$$R_{t+1} = r(S_t, S_{t+1}, A(t))$$

- **Bellman equations** for the state value function:

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} p(s' | s, \pi(s))(r(s, s', \pi(s)) + \gamma V^\pi(s'))$$

# Value function computation example



- Policy:  $\pi(\text{high}) = \text{search}$ ,  $\pi(\text{low}) = \text{search}$
- Bellman equations:

$$V^\pi(\text{h}) = \alpha(r_s + \gamma V^\pi(\text{h})) + (1 - \alpha)(r_s + \gamma V^\pi(\text{l}))$$

$$V^\pi(\text{l}) = (1 - \beta)(-3 + \gamma V^\pi(\text{h})) + \beta(r_s + \gamma V^\pi(\text{l}))$$

# Solving the linear system

$$V^\pi(h) = \alpha(r_s + \gamma V^\pi(h)) + (1 - \alpha)(r_s + \gamma V^\pi(1))$$

$$V^\pi(1) = (1 - \beta)(-3 + \gamma V^\pi(h)) + \beta(r_s + \gamma V^\pi(1))$$

Rewrite in “linear algebra” style:

$$(I - \gamma P)V = b$$

Where:

$$P = \begin{bmatrix} \alpha & 1 - \alpha \\ 1 - \beta & \beta \end{bmatrix}, \quad \begin{bmatrix} r_s \\ -3(1 - \beta) + r_s\beta \end{bmatrix}$$

# Numerical solution

Parameters:

$$\alpha = 0.8, \quad \beta = 0.3$$

$$r_{\text{search}} = 15, \quad r_{\text{wait}} = 10, \quad r_{\text{empty}} = -3.0$$

For  $\gamma = 0.9$ :

$$A = I - \gamma P = \begin{bmatrix} 0.28 & -0.18 \\ -0.63 & 0.73 \end{bmatrix}, \quad b = \begin{bmatrix} 15. \\ 2.4 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} V(\text{high}) \\ V(\text{low}) \end{bmatrix} = \begin{bmatrix} 125.07692308 \\ 111.23076923 \end{bmatrix}$$