SMU Lyle School of Engineering

Reinforcement Learning

Markov Decision Processes (MDP)

Sutton/Barto* Chapter 3

Michael Hahsler, SMU

With figures from Sutton/Barto*

*Sutton and Barto, Reinforcement Learning: An Introduction, 2nd edition, MIT Press, Cambridge, MA, 2018

Topics of this Course

- Introduction to reinforcement learning
- **Markov decision processes**
- Part I: Tabular Methods
  - Dynamic programming
  - Monte Carlo methods
  - Temporal-difference learning
  - Multi-step bootstrapping
  - Planning and learning with tabular methods
- Part II: Approximate Solution Methods
  - Prediction and Control using Approximation
  - Eligibility Traces
  - Policy Gradient Methods
- Part III: Modern RL Methods
  - Deep Reinforcement Learning
  - Current Applications

Summary of Notation

General

X	capital letters: random variables
x, p	lower-case letters: realizations of random variables or scalar functions
$\boldsymbol{w}$	Bold lower-case letters: real-valued vectors (even if random variables)
$\boldsymbol{W}$	bold capitals: matrices
α	Greek letters: parameters (vectors if in bolt)
$\Pr\{X = x\}$	probability that a random variable X takes on the value x
$X \sim p$	random variable X selected from distribution $p(x) = \Pr\{X = x\}$
$\mathbb{E}[X]$	expectation of a random variable X, i.e., $\mathbb{E}[X] = \sum_x p(x)x$
$\text{argmax}_a f(a)$	a value of action a at which $f(a)$ takes its maximal value

Value Function

G_t	return (cumulative reward) following time t
$G_{t:h}$	return from t to h (discounted and corrected)
$v_\pi(s)$	value of state s under policy π (expected return)
$v_*(s)$	value of state s under the optimal policy
$q_\pi(s, a)$	value of taking action a in state s under policy π
$q_*(s, a)$	value of taking action a in state s under the optimal policy
V, V_t	array estimates of state-value function v_π or v_*
Q, Q_t	array estimates of action-value function q_π or v_*

MDP

s, s'	states
a	an action
r	a reward
$\mathcal{S}$	set of all (nonterminal) states, $\mathcal{S}^+$ are all states
$\mathcal{A}(s)$	set of all actions available in state s
γ	discount-rate parameter
t	discrete time step
T	final time step of an episode (a.k.a. horizon)
A_t	random variable for the action at time t
S_t	random variable for the state at time t
R_t	random variable for the reward at time t
$p(s', r \mid s, a)$	probability of transition to state s' and receiving reward r, from state s taking action a.
$p(s' \mid s, a)$	probability of transition to state s' fom state s taking action a.
$r(s, a)$	expected immediate reward from state s after action a.
$r(s, a, s')$	expected immediate reward from state s to s' with action a.
$\pi(a \mid s)$	probability of taking action a in state s under stochastic policy π
$\pi(s)$	action taken in state s under deterministic policy π

Finite Markov Decision Processes

The foundation of RL

The Agent–Environment Interface

- Markov decision process (MDP) is a mathematical model **for sequential decision making** when outcomes are uncertain.
- The MDP represents a **fully observable, stochastic, and known environment.**
- The goal is specified via rewards.
- Actions influence the future rewards by leading to different states.

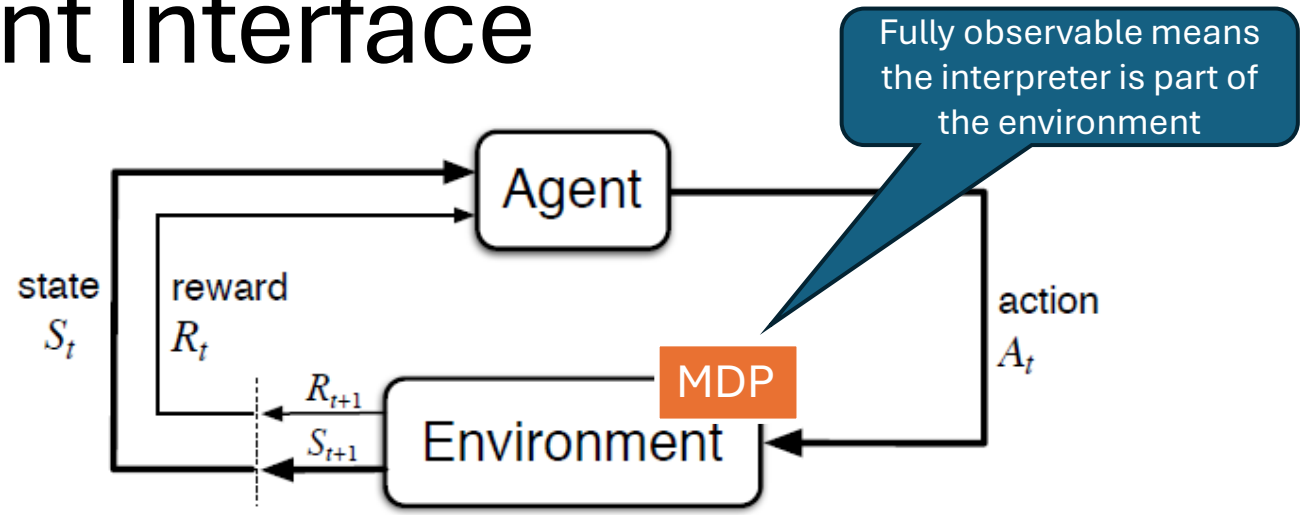


Figure 3.1: The agent–environment interaction in a Markov decision process.

We consider **discrete-time MDPs** where the **interaction** is a sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$

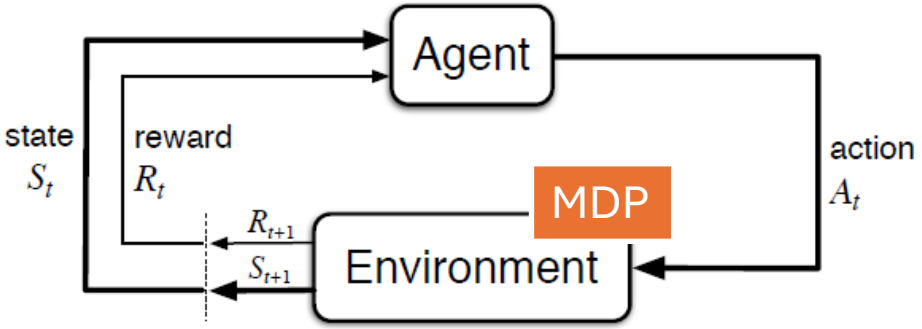
- At each time step t:
  1. Agent receives some representation of the environment's state $s_t \in \mathcal{S}$
  2. Agent selects an action, $a_t \in \mathcal{A}(s_t)$
  3. One time step later (as a consequence of its action) the agent receives a numerical reward, $r_{t+1} \in \mathcal{R}$ and finds itself in a new state s_{t+1}

Leads to sequence or trajectory that begins like this:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

Finite MDP

- General MDPs can have continuous state and action spaces.
- We consider finite MDPs. An MDP is **finite** if the
  - sets of states $\mathcal{S}$,
  - set of action $\mathcal{A}$, and
  - set of rewards $\mathcal{R}$ all have a finite number of elements.
  - It always has discrete time steps.



- The **dynamics** of the finite discrete-time MDP are described by the function $p: \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S} \rightarrow [0,1]$ defined as

$$p(s', r|s, a) \stackrel{\text{def}}{=} \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- A decision process is an MDP if p completely characterizes the environment's dynamics. This implies:
  - Dynamics only depend on the current observed state s.
  - This means: the state must have sufficient information about the past.
  - If this is true, then it is called an **information state** and the system has the necessary **Markov property**:

$$\Pr\{s_{t+1}|\, s_t, a_t\} = \Pr\{s_{t+1}|\, s_0, a_0, s_1, a_1, \ldots, s_t, a_t\}$$

Note: When we say MDP in this class, we will always mean a finite discrete-time MDP.

Transition and Reward Functions

- Often the function $p(s', r|s, a)$ is broken down into a transition and a reward model.

- State transition probabilities:

$$p(s'|s, a) \stackrel{\text{def}}{=} \Pr\{S_t = s', \mid S_{t-1} = s, A_{t-1} = a\}$$

$$= \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

- Expected reward for state-action pairs:

$$r(s, a) \stackrel{\text{def}}{=} \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a]$$

$$= \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

- Expected rewards for state-action-next-state triples (if the next state also affects the reward):

$$r(s, a, s') \stackrel{\text{def}}{=} \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s']$$

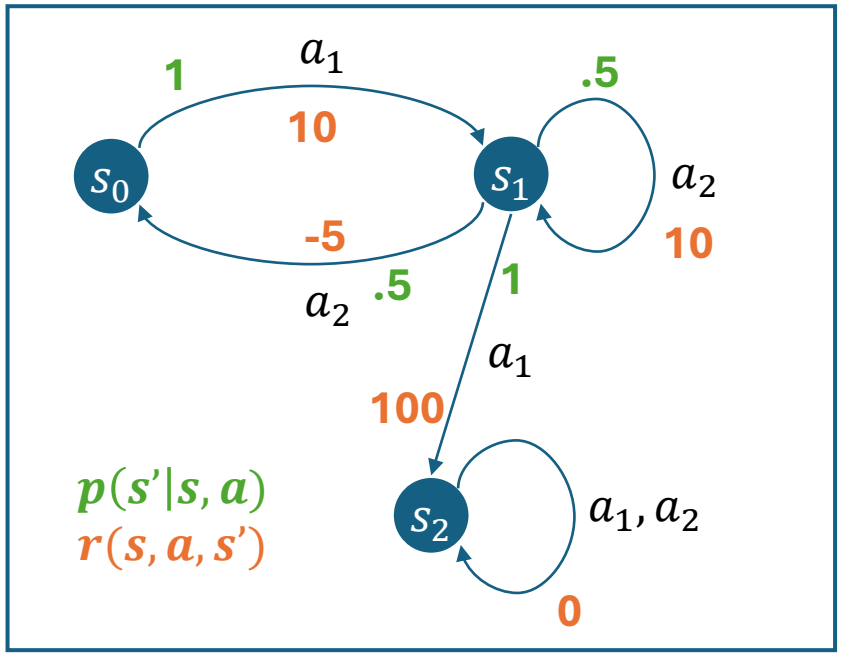
$$= \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

Finite Markov Decision Process (MDP)

Finite MDPs are discrete-time stochastic control processes defined by:

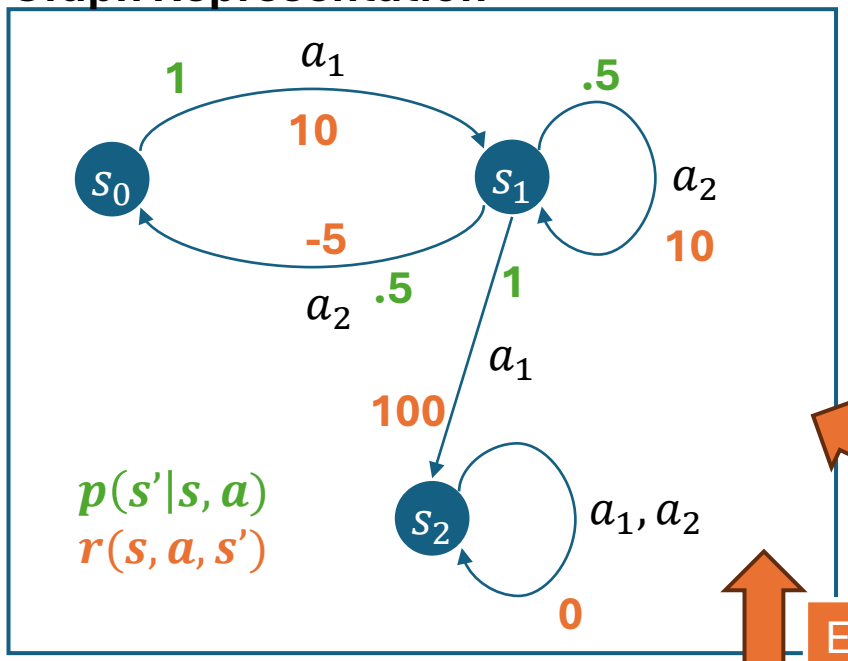
- a finite set of **states** $\mathcal{S} = \{s_0, s_1, s_2, \dots\}$ (initial state s_0)
- a set of available **actions** $\mathcal{A}(s)$ in each state s
- a **transition model** $p(s' \mid s, a)$ where $a \in \mathcal{A}(s)$
- a **reward function** $r(s)$ where the immediate reward depends on the current state (often $r(s, a, s')$ is used to make modelling easier)

Example MDP as a Graph



Alternative Representations of a Finite MDP

Graph Representation



p-Function Table Representation

s	a	s'	$p(s'\|s,a)$	$r(s,a,s')$
s_0	a_1	s_1	1	10
s_1	a_2	s_0	.5	-5
s_1	a_2	s_1	.5	10
s_1	a_1	s_1	1	100
s_2	a_1	s_2	1	0
s_2	a_2	s_2	1	0

Equivalent

Matrix Representation

Transition matrices

a_1

	s_0	s_1	s_2
s_0	0	1	0
s_1	0	0	1
s_2	0	0	1

a_2

	s_0	s_1	s_2
s_0	1^*	0^*	0^*
s_1	.5	.5	0
s_2	0	0	1

Reward matrices

a_1

	s_0	s_1	s_2
s_0	$-\infty$	10	$-\infty$
s_1	$-\infty$	$-\infty$	100
s_2	$-\infty$	$-\infty$	0

a_2

	s_0	s_1	s_2
s_0	$-\infty$	$-\infty$	$-\infty$
s_1	-5	10	$-\infty$
s_2	$-\infty$	$-\infty$	0

* illegal actions, but the rows in the transition matrices need to add up to 1!
$-\infty$ represents illegal actions and transitions in the reward matrices!

Example: Mobile Robot MDP

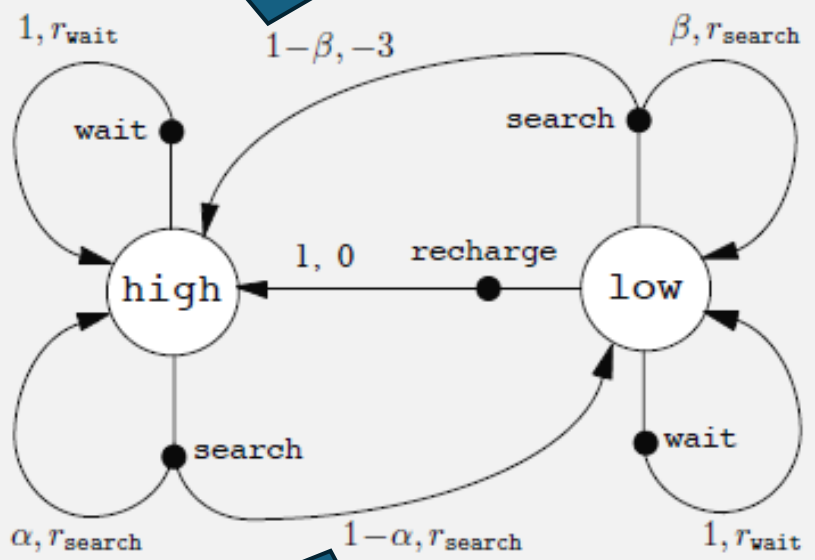
A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. It can actively search for cans or wait for people to drop them off.

States: $\mathcal{S} = \{\text{high}, \text{low}\}$ // battery charge

Actions: $\mathcal{A} = \{\text{search}, \text{wait}, \text{recharge}\}$

Reward: each collected can gives +1

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	r_{search}
low	search	high	$1 - \beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-

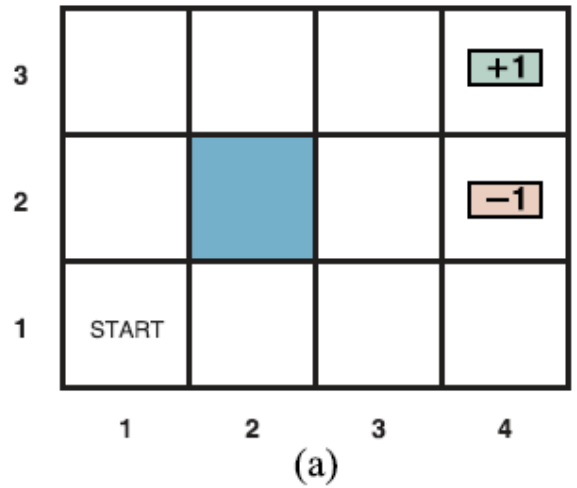


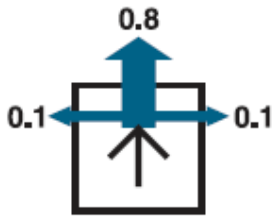
$1 - \beta$: Chance of running out of battery and needing rescue

Expected number of cans collected.

$1 - \alpha$: Battery runs low

Exercise: 3x4 Grid World





AIMA Figure 17.1: (a) A simple, stochastic environment. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. Transitions into the two terminal states have reward +1 and -1, respectively, and all other transitions have a reward of -0.04.

State representation: (x, y)

1. Draw MDP model as a graph.

2. Complete the following entries in the model as a table

s	a	s'	$p(s'\|s,a)$	$r(s,a,s')$
(1,1)	up	(1,2)		
(1,1)	right	(1,2)		
(1,2)	up	(1,3)		
(3,3)	right	(4,3)		
(3,3)	right	(1,1)		

Rewards and Returns

What is the agent’s goal?

Reward Signals

- The reward signal gives the agent feedback on how well it is doing.

- The reward signal at time t is a single number from a finite set:

$$R_t \in \mathcal{R} \subseteq \mathbb{R}$$

- Rewards communicate
  - **What** the agent needs to achieve, and
  - **not how** to do it. That is what the agent should learn!

- Options
  - Only provide a reward when the agent finishes the task. This is very clear, but delayed rewards may make learning more difficult.
  - **Reward shaping**: Providing intermediate rewards for partial task completion can improve learning, but may lead to issues like reward hacking.

Return and Episodes

- The agent's objective is to maximize the return of an episode.
- The return is defined as

$$G_t \stackrel{\text{def}}{=} R_{t+1} + R_{t+2} + R_{t+3} + \cdots R_T$$

  where T is the final step in the episode.

We use G so the return is not confused with rewards.

- Episodic task:
  - Episodes end with a terminal state at time T.
  - Typically followed by a reset for the next episode.

- Continuing task:
  - Has no episodes! $T = \infty$

- **Note**: An episodic task can be modeled as a continuing task with all rewards after T set to 0. Typically, the state reached a time T is modeled as an absorbing state.

Reward Discounting

- Discounting is often used when rewards now are more important then rewards later. Continuing tasks always use discounting.

$$G_t \stackrel{\text{def}}{=} R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

  where $0 \leq \gamma < 1$ is the discount factor.

- Properties:
  - Discounting guarantees a finite sum for G_t if $\{R_k\}$ is bounded for continuing tasks
  - "myopic" agents use $\gamma = 0$ and only maximizes immediate rewards.
  - Larger γ make the agent more farsighted.

- Return is recursive:

$$\begin{aligned} G_t &\stackrel{\text{def}}{=} R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= R_{t+1}\, \gamma\, (R_{t+2} + R_{t+3} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Policy and Value Functions

How should an agent act?

Policy

A (Stochastic) **Policy**: a mapping from states to probabilities of selecting each possible action.

$$\pi(a|s)$$

- Following policy π at time t means to chose action a in state s with probability $\pi(a|s)$

Soft policies

- A stochastic policy is soft if all $\pi(a|s) > 0$.
  I.e., the policy will eventually try all actions.

Stochastic Policy as a Table: $\pi(a|s)$

s	up	right	down	left
(1,1)	0.6	0.1	0.1	0.1
…	…	…	…	…
…	…	…	…	…

Deterministic Policy

Deterministic policy: Prescribes for each state a single action.

$$a = \pi(s)$$

- This just means we have a stochastic policy that has a probability of 1 for exactly one action.

- **Important guarantee**: Finite MDPs always have an optimal deterministic policy.
- **Intuitive reason**: If there is a best action given all we know (i.e., we know the current state), then there is no point in choosing a suboptimal action.

Deterministic Policy as a Table

s	Action $\pi(s)$
(1,1)	up
…	…
…	…

Stochastic Policy as a Table: $\pi(a|s)$

	up	right	down	left
(1,1)	1	0	0	0
…	…	…	…	…
…	…	…	…	…

Value Function

- A policy $\boldsymbol{\pi}$ defines for each state which action to take.
- **State value function**: Expected value of being in a state when following π.

$$v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right] \text{ for all } s \in \mathcal{S}$$

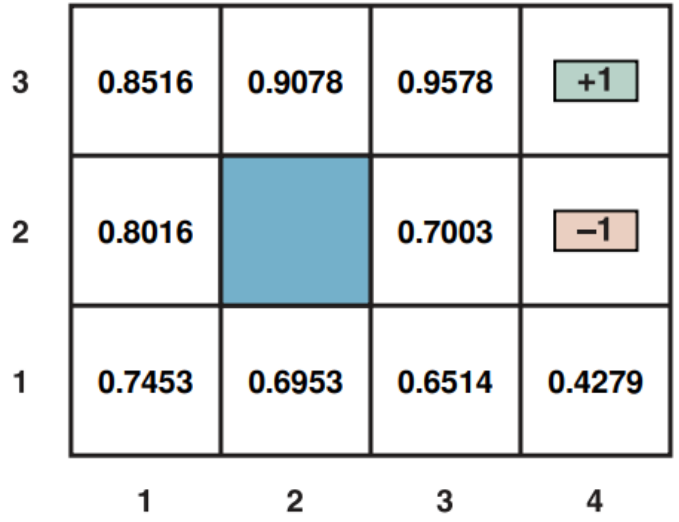
- Representation as a tabular estimate V:

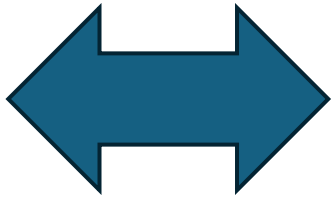
γ .. Discounting factor to give more weight to immediate rewards.

$\mathbb{E}_\pi$ ... Expectation over sequences that can be created by following π.

R_t .. Reward at time t

Value Function for the 3x4 Grid World





Value Function

s	State Value $V(s)$
(1,1)	0.7453
(1,2)	0.8016
...	...

Action-value Function: Q-function

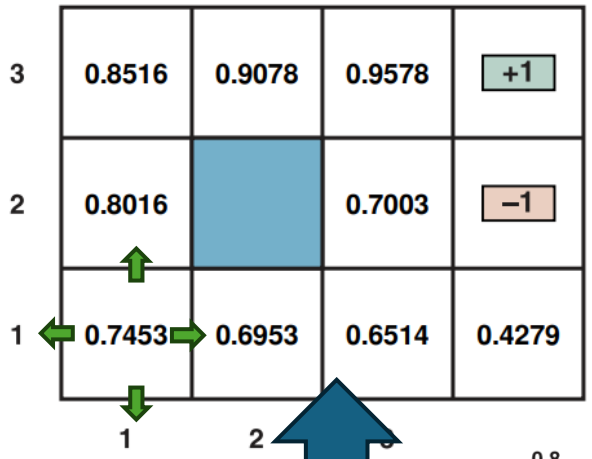
- Expected value of choosing an action in a state and then following π.

$$q_\pi(s,a) \overset{\text{def}}{=} \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \quad \text{for all } s \in \mathcal{S}$$

- The Q-function lets us compare the value of taking different action is a given state. It is used in algorithms to determine what action is the best (the action with the highest Q-value for the state).

Value Function



Q-Table

s	a	$q(s,a)$
(1,1)	up	0.7453
(1,1)	right	0.6709
(1,1)	down	0.7003
(1,1)	left	0.7109
…	…	…

Value Function Estimation using Monte Carlo Methods

- The value functions v_π and q_π can be estimated from experience. This is called **policy evaluation**.

- A simple Monte Carlo method: **Sample-average method**:
  - Follow policy π.
  - Maintains an average $\bar{V}(s)$ of the actual returns that have followed that state. The average is updated at each visit.
  - Convergence: $\lim_{visits \to \infty} \bar{V}(s) = v_\pi(s)$

- This also works for $q_\pi(\mathrm{s}, \mathrm{a})$ with $\lim_{visits \to \infty} \bar{Q}(s, a) = q_\pi(s, a)$

- For problems with many states: Instead of storing all $\bar{V}(s)$/ $\bar{Q}(s, a)$ in a table, learn an **approximate function** with a small number of parameters.

The Optimal Policy

- Value functions define a partial ordering over policies.
  A policy is better if it has at least one larger state value.

$$\pi' \geq \pi \Leftrightarrow v_{\pi'}(s) \geq v_{\pi}(s) \text{ for all } s \in \mathcal{S}$$

- There always exists an **optimal policy** π_* with the optimal state-value function v_* defined as

$$v_*(s) \stackrel{\text{def}}{=} \max_{\pi} v_{\pi}(s) \text{ for all } s \in \mathcal{S}$$

- There also exists an optimal action-value function associated with the optimal π_*:

$$q_*(s,a) \stackrel{\text{def}}{=} \max_{\pi} q_{\pi}(s,a) \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}$$

- Determining the optimal policy:
  - It is optimal to always choose this best action.
  - If we know all $q_*(s,a)$ values, then we can determine the best action for each state by the highest Q-value:

$$\pi_*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} q_*(s,a)$$

  - This is called a greedy policy (with respect to q) and it is deterministic.
    MDPs always have an optimal deterministic policy!

Code...

The Bellman Equations

Bellman’s recursive formulation of value functions.

Bellman Expectation Equations

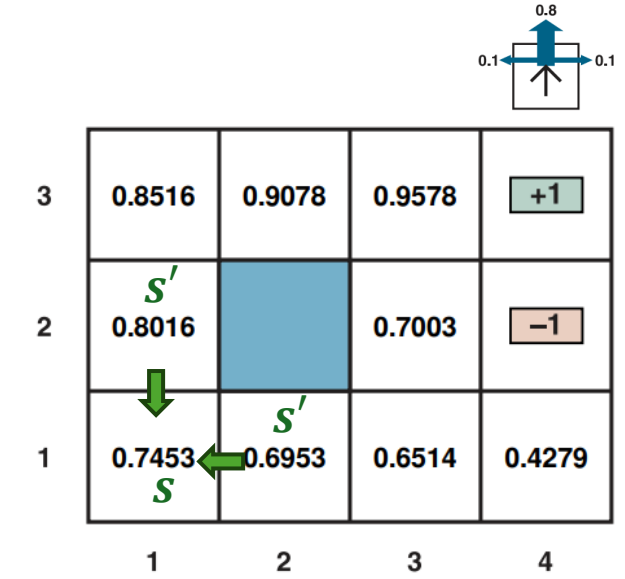
The state values **given a policy π** are defined by the expected return of following the policy:

$$v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t|S_t = s] \qquad \forall s \in \mathcal{S}$$

Requires information about complete episodes.

Bellman showed that the value function has a **recursive relationship:**

$$\begin{aligned} v_\pi(s) &\stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned}$$



Bellman Expectation Equations

$$v_\pi(s) \stackrel{\text{def}}{=} \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \text{ for all } s \in \mathcal{S}$$

Alternative formulation with transition and reward function:

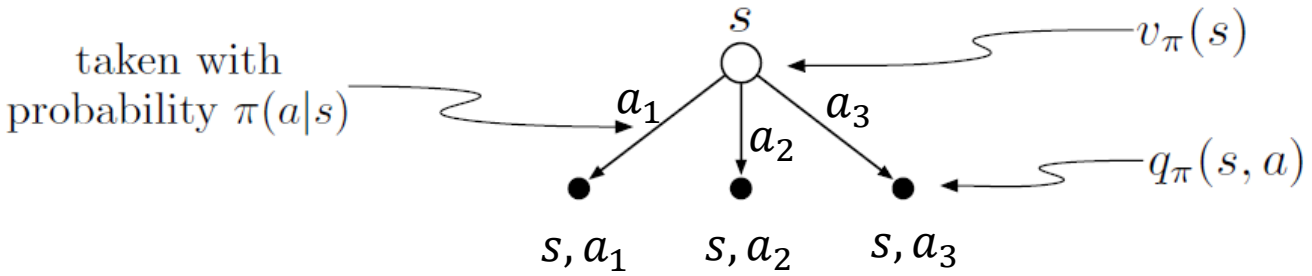
$$v_\pi(s) \stackrel{\text{def}}{=} \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v_\pi(s')] \text{ for all } s \in \mathcal{S}$$

Transforms the problem of finding returns into a problem of finding **local consistency** between state values.

Backup Diagrams: Notation

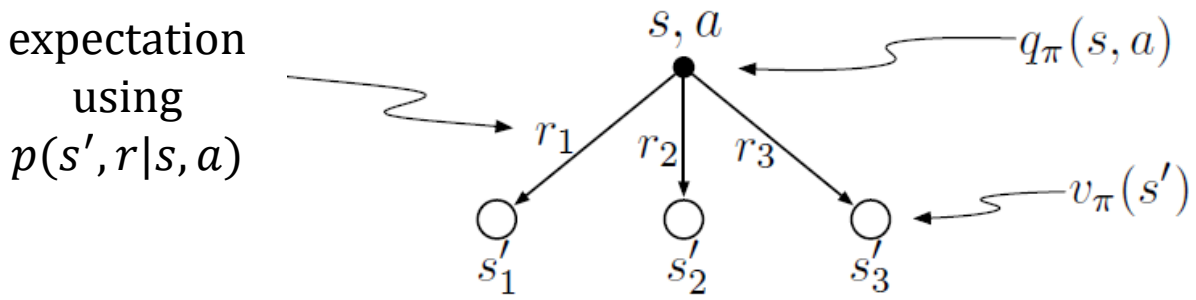
- The relationship between state values, action probabilities, state-action values and expected rewards can be visualized as a tree.

Choosing an action in a state (= calculate the expectation over actions)



$$v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$$

Calculate the expected state-action value



$$q_\pi(s,a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

Bellman Equation as a Backup Diagram

$$v_\pi(s) \stackrel{\text{def}}{=} \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \text{ for all } s \in \mathcal{S}$$

Expectation

Choose action

Expectation for transition and reward

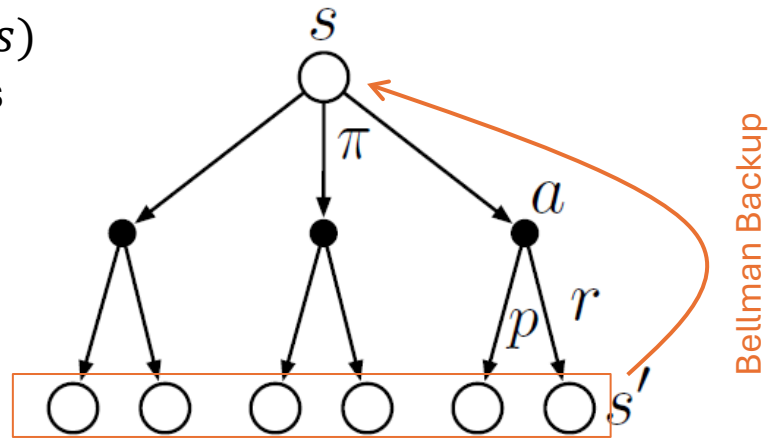
Reward

Value of next state

Backup diagram for $v_\pi(s)$ shows all possible paths for the expectation.

All paths are of the form:

s, a, r, s'



Bellman Optimality Equations

- The **optimal value function** is defined the value function of the optimal policy:

$$v_*(s) \stackrel{\text{def}}{=} \max_\pi v_\pi(s) \qquad \text{for all } s \in \mathcal{S}$$

- We can express the optimal policy by the greedy actions that it chooses and then apply Bellman's recursive relation between state values:

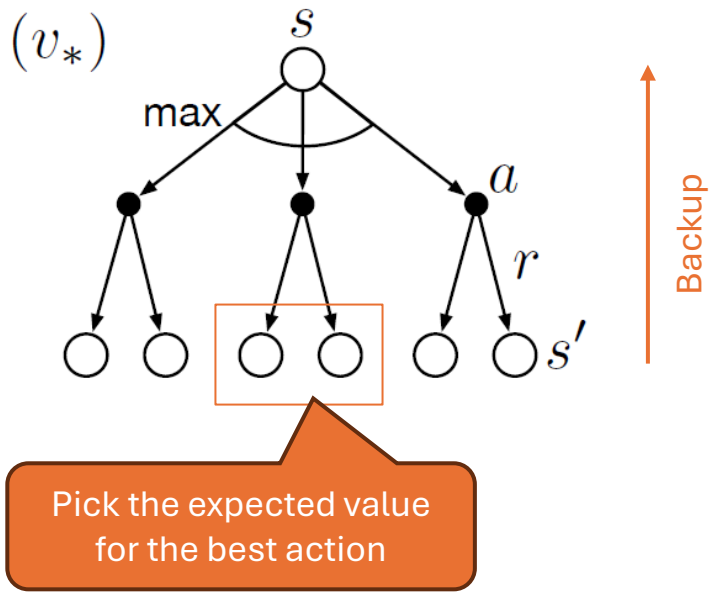
$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \end{aligned}$$

Bellman Optimality Equations

$$v_*(s) = \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')] \qquad \text{for all } s \in \mathcal{S}$$

Advantage: Expresses optimality as local consistency between state values without:
- A need for complete episodes
- An explicit policy in the equation

Backup diagram with max operator



Solving the Bellman Optimality Equations

- The Bellman optimality equations form a system of $|\mathcal{S}|$ non-linear equations of the from:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s,a) \quad \text{for all } s \in \mathcal{S}$$

max() is a non-linear operator.

- **Issues** with solving the system:
  - The **Markov property** for states has to be satisfied.
  - The **transition and reward model** (function p) needs to be completely known.
  - Computational resources for **solving a large system of non-linear equations.**

- This is rarely possible in real applications and we need approximate methods.

Relationship between Value Functions and Policies

Value functions are defined as expected returns following a policy:

$$v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t|S_t = s]$$

$$q_\pi(s,a) \stackrel{\text{def}}{=} \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

For the optimal policy and value function we have:

$$q_*(s,a) \stackrel{\text{def}}{=} \mathbb{E}_{\pi^*}[G_t|S_t = s, A_t = a]$$

$$v_*(s) = \max_{a\in\mathcal{A}} q_*(s,a)$$

$$\pi^*(s) = \operatorname*{argmax}_{a\in\mathcal{A}} q_*(s,a)$$

Recursively defining G_t as $R_{t+1} + \gamma v(S_{t+1})$ leads to:

Bellman Expectation Equations

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$

Bellman Optimality Equations

$$v_*(s) = \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a]$$

Related Models

Goal-based Agents, Simplifications, and Extensions

Learning With an MDP vs. Goal-based Agent

MDP

- **Objective**: Maximize the episode return.
  - Goal states do not exist.
  - The "goal" is defined by the reward structure.
  - Absorbing (terminal) states mean that no more reward can be accumulated.
- **Solution method**: learn a policy that maximizes the expected return.

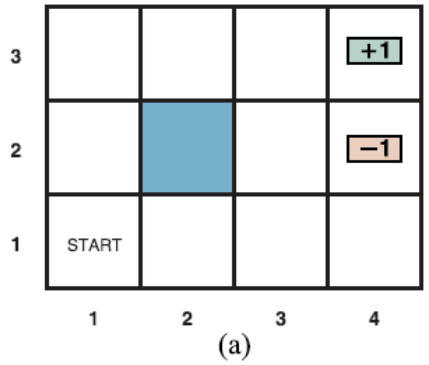
Goal-based Agent

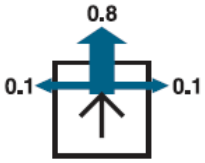
- **Objective**: reach a goal state.
- **Solution method**: searches for the shortest path.

Create goal-based behavior using an MDP:

- Making the "goal state" a terminal state.
- Only awarding a reward when the agent enters that state.

MDPs are much more general and can easily incorporate "goal states" with different values and stochastic environments (transition models).





(b)

Related Models

MDP simplifications:

- **Multi-armed bandits (Chapter 2 in Sutton/Barto)**
  - You repeatedly choose an arm (action).
  - You immediately get a reward and start over.
  - There is no state evolution.
  - Examples: flipping a coin repeatedly, playing multiple slot machines

- **Markov Reward Process**: A Markov chain with rewards (no actions).

MDP extension:

- Partially Observable Decision Process (POMDP): a MDP for partially observable problems. It adds:
  - Observations O
  - Observation probabilities $Z(o|s,a)$

Many other related models were developed: Semi-Markov decision processes, contextual bandits, Markov games, constrained MDPs, multi-objective MDPs, Dynamic Bayesian Networks

Partially Observable Markov Decision Process (POMDP)

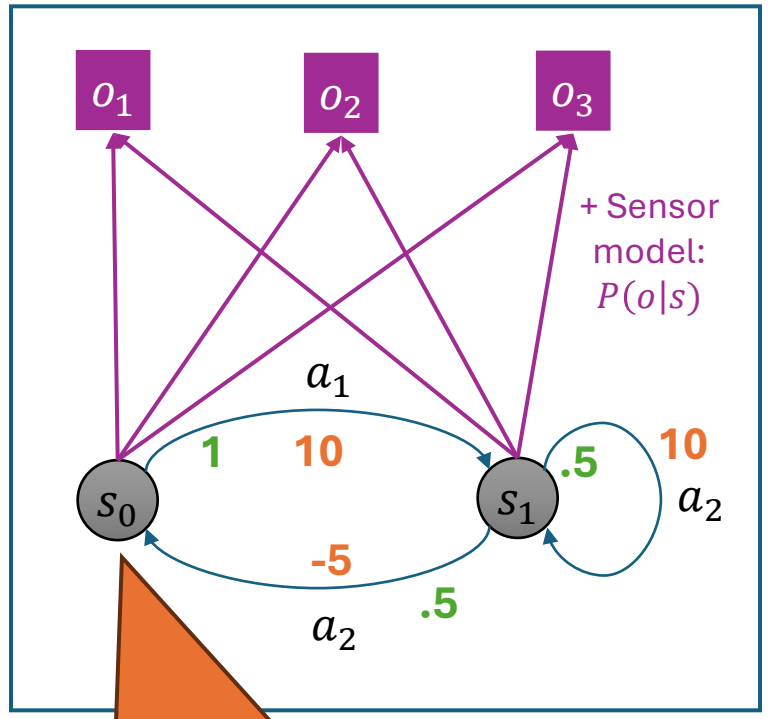
- MDPs assume full observability, which is a very unrealistic assumption!
  MDPs can be extended for partially observable environments. The agent observes part of the environment state as an observation o_t ($\boldsymbol{x}_t$ is also commonly used).
- **State Estimation**: The agent tries to reconstruct the complete state information using all available information (actions and observations):

$$\widehat{s_t^e} = \text{update}\left(\text{predict}\left(\widehat{s_{t-1}^e}, a_t\right), o_t\right)$$

- Prediction is done using the MPD's transition model, the update (filtering) is performed using an additionally known **sensor model** specifying $P(o|s)$, the probability for receiving observation o given being in state s.
- **Belief states**: The estimate is a belief state expressed as a distribution over all states. It updated using the Bayesian rule when new observations become available.

  Example: For a problem with three states, the belief state $b = (.2, .8, 0)$ means the agent believes that it is with 20% in state 1 and 80% in state 2, but not in state 3. If it observes something that indicates that it is more likely in state 1 then it may update the believe state to $b = (.3, .7, 0)$.
- **Belief MDP:** A POMDP can be converted into a MDP that uses belief states instead of system states.

  Issue: The probabilities in belief states are continuous, leading to a continuous (infinite) state space. The belief MDP is not a finite MDP!
- **Policy**: The solution of a belief MDP/POMDP is a policy with the optimal actions for sets of belief states (i.e., ranges of belief).
- **Complexity**: For all but tiny problems, POMDPs can only be solved **approximately** (e.g., by grid-based methods).



The agent knows the transition model and the sensor model, but is cannot observe what stat it the environment is in.

Summary: What You Should Know

Terms

- **Actions** are the agent's choices
- **States** are the basis for the choices
- **Reward** is used to evaluate choices
- **Objective**: maximize expected return
- **Policy**: a stochastic rule to select actions as a function of the state
- **Information state** = **Markov property** for states

Backup Diagrams to represent the calculation of expected values.

Bellman optimality Equations

- **Recursive relationship** between state values.
- MDPs always have an **optimal deterministic policy**.

Tasks we will talk in the next sessions:

- **Policy evaluation** (prediction): estimate v_π or q_π for a given fixed π.
- **Control**: find optimal π_*