MDPs: overview

Markov decision process

Definition: Markov decision process

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

$T(s'|s,a)$: probability of s' if take action a in state s

Reward(s,a,s'): reward for the transition (s,a,s')

IsEnd(s): whether at end

$0 \leq \gamma \leq 1$: discount factor (default: 1)

What is a solution?

Search problem: path (sequence of actions)

MDP:

Definition: policy

A **policy** π is a mapping from each state $s \in$ States to an action $a \in$ Actions(s).

Example: volcano crossing

s	$\pi(s)$
(1,1)	S
(2,1)	E
(3,1)	N
...	...

MDPs: policy evaluation

Discounting

Definition: utility

Path: $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \dots$ (action, reward, new state).

The **utility** with discount γ is

$$u_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$$

Discount $\gamma = 1$ (save for the future):

[stay, stay, stay, stay]: $4 + 4 + 4 + 4 = 16$

Discount $\gamma = 0$ (live in the moment):

[stay, stay, stay, stay]: $4 + 0 \cdot (4 + \cdots) = 4$

Discount $\gamma = 0.5$ (balanced life):

[stay, stay, stay, stay]: $4 + \frac{1}{2} \cdot 4 + \frac{1}{4} \cdot 4 + \frac{1}{8} \cdot 4 = 7.5$

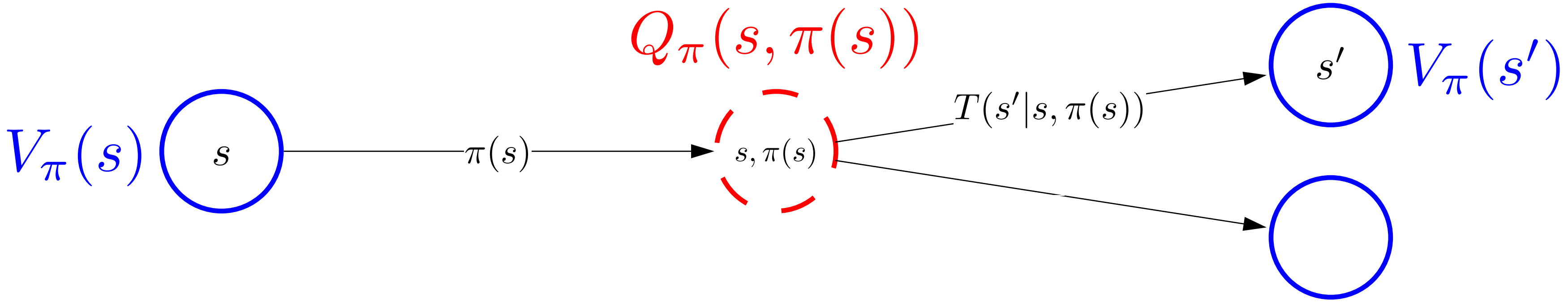
Policy evaluation

Definition: value of a policy

Let $V_\pi(s)$ be the expected utility received by following policy π from state s.

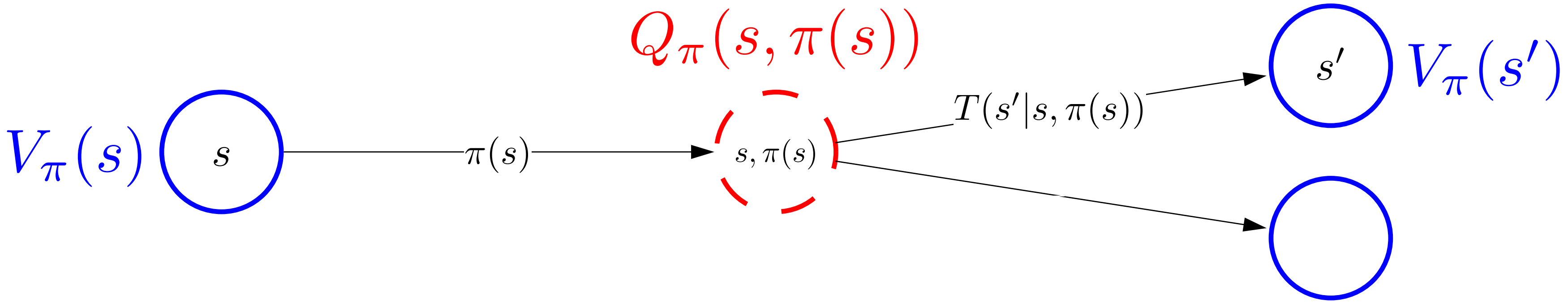
Definition: Q-value of a policy

Let $Q_\pi(s, a)$ be the expected utility of taking action a from state s, and then following policy π.



Policy evaluation

Plan: define recurrences relating value and Q-value



$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s, \pi(s)) & \text{otherwise.} \end{cases}$$

$$Q_\pi(s, a) = \sum_{s'} T(s'|s, a)[\text{Reward}(s, a, s') + \gamma V_\pi(s')]$$

Policy evaluation

Key idea: iterative algorithm

Start with arbitrary policy values and repeatedly apply recurrences to converge to true values.

Algorithm: policy evaluation

Initialize $V_\pi^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \dots, t_{\text{PE}}$:

For each state s:

$$V_\pi^{(t)}(s) \leftarrow \underbrace{\sum_{s'} T(s'|s,\pi(s))[\text{Reward}(s,\pi(s),s') + \gamma V_\pi^{(t-1)}(s')]}_{Q^{(t-1)}(s,\pi(s))}$$

MDPs: value iteration

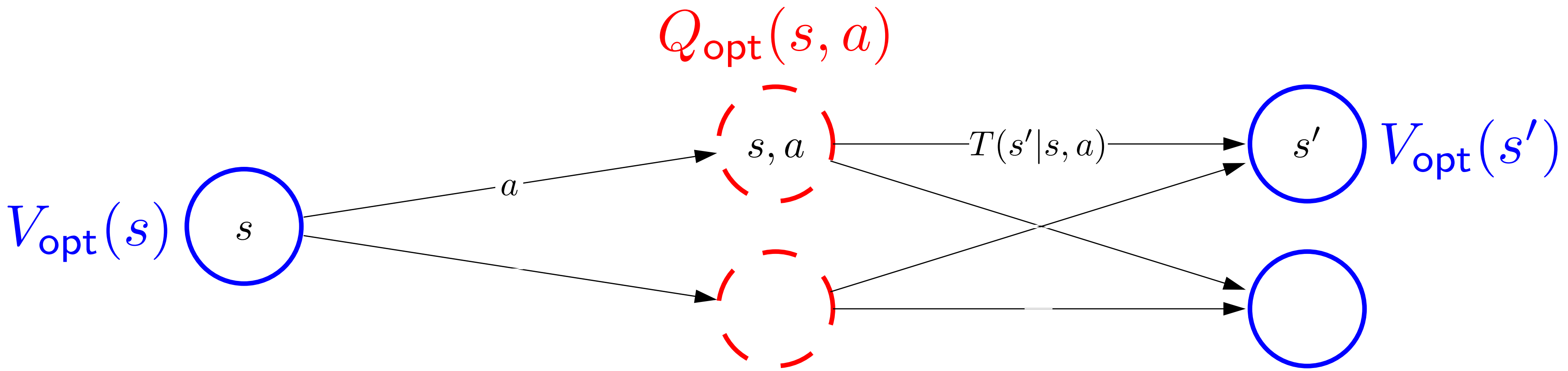
Optimal value and policy

Goal: try to get directly at maximum expected utility

Definition: optimal value

The **optimal value** $V_{\text{opt}}(s)$ is the maximum value attained by any policy.

Optimal values and Q-values



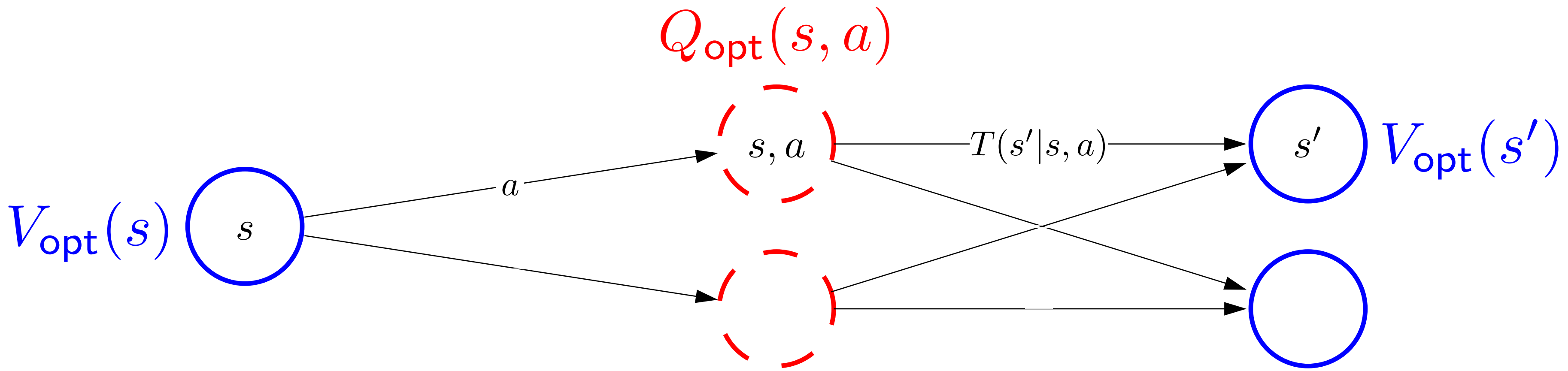
Optimal value if take action a in state s:

$$Q_{\text{opt}}(s, a) = \sum_{s'} T(s, a, s')[\text{Reward}(s, a, s') + \gamma V_{\text{opt}}(s')].$$

Optimal value from state s:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s, a) & \text{otherwise.} \end{cases}$$

Optimal policies



Given Q_{opt}, read off the optimal policy:

$$\pi_{\text{opt}}(s) = \arg \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s,a)$$

Value iteration

Algorithm: value iteration [Bellman, 1957]

Initialize $V_{\text{opt}}^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \dots, t_{\text{VI}}$:

  For each state s:

$$V_{\text{opt}}^{(t)}(s) \leftarrow \max_{a \in \text{Actions}(s)} \underbrace{\sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}^{(t-1)}(s')]}_{Q_{\text{opt}}^{(t-1)}(s,a)}$$

Time: $O(t_{\text{VI}} S A S')$

Convergence

Theorem: convergence

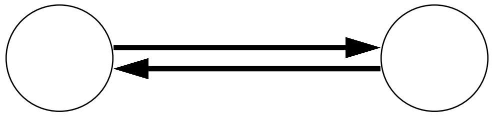
Suppose either

- discount $\gamma < 1$, or
- MDP graph is acyclic.

Then value iteration converges to the correct answer.

Example: non-convergence

discount $\gamma = 1$, zero rewards



Summary of algorithms

- Policy evaluation: (MDP, π) $\to V_\pi$

- Value iteration: MDP $\to (Q_{\text{opt}}, \pi_{\text{opt}})$

MDPs: reinforcement learning

Unknown transitions and rewards

Definition: Markov decision process

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

IsEnd(s): whether at end of game

$0 \leq \gamma \leq 1$: discount factor (default: 1)

reinforcement learning!

MDPs: model-based methods

Model-Based Value Iteration

Data: $s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \dots; a_n, r_n, s_n$

Key idea: model-based learning

Estimate the MDP: $T(s'|s,a)$ and $\text{Reward}(s,a,s')$

Transitions:

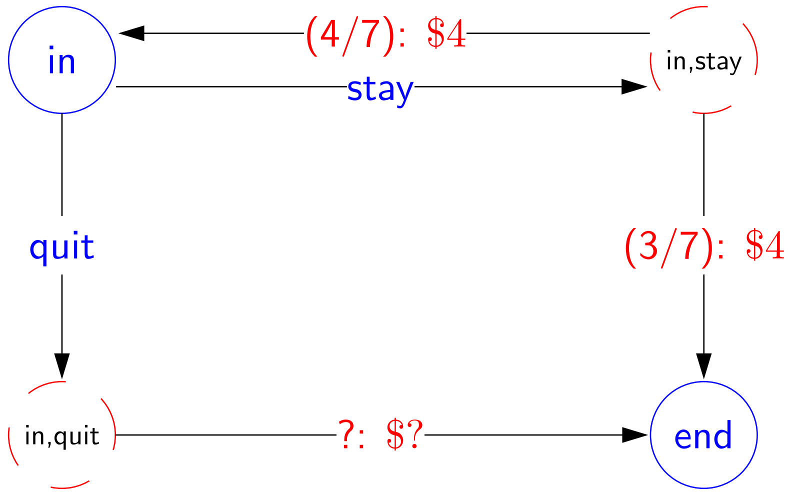
$$\hat{T}(s'|s,a) = \frac{\#\text{ times } (s,a,s') \text{ occurs}}{\#\text{ times } (s,a) \text{ occurs}}$$

Rewards:

$$\widehat{\text{Reward}}(s,a,s') = r \text{ in } (s,a,r,s')$$

Compute policy using value iteration under estimated MDP $(\hat{T}, \widehat{\text{Reward}})$.

Model-Based Value Iteration

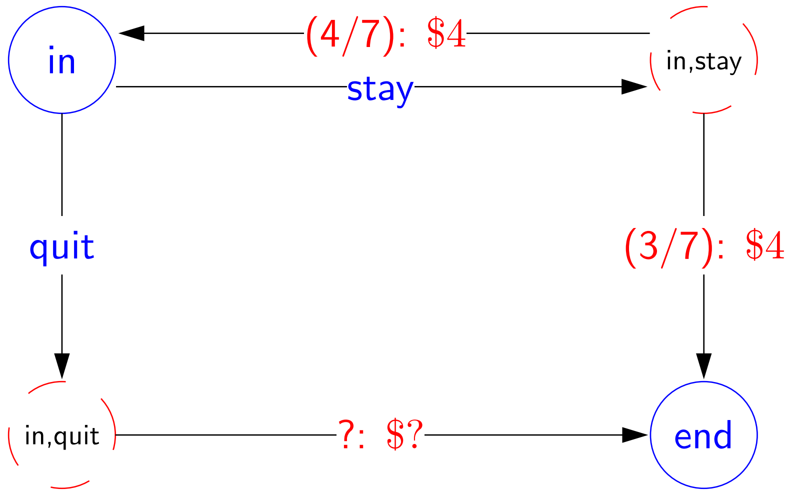


Data (following policy $\pi(s) =$ stay):

[in; stay, 4, end]

- Estimates converge to true values (under certain conditions)
- With estimated MDP $(\hat{T}, \widehat{\text{Reward}})$, compute policy using value iteration

Problem



Problem: won't even see (s, a) if $a \neq \pi(s)$ ($a =$ quit)

Key idea: exploration

To do reinforcement learning, need to explore the state space.

Solution: need π to **explore** explicitly (more on this later)

MDPs: model-free methods

From model-based to model-free

$$\hat{Q}_{\text{opt}}(s,a) = \sum_{s'} \hat{T}(s'|s,a)[\widehat{\text{Reward}}(s,a,s') + \gamma \hat{V}_{\text{opt}}(s')]$$

All that matters for policy learning is the estimate of $Q_{\text{opt}}(s,a)$.

Key idea: model-free reinforcement learning

Try to estimate $Q_{\text{opt}}(s,a)$ directly.

This module: start by estimating Q_π.

Model-free Monte Carlo

Data (following policy π):

$$s_0; a_1, r_1, s_1; a_2, r_2, s_2; a_3, r_3, s_3; \ldots; a_n, r_n, s_n$$

Recall:

$Q_\pi(s, a)$ is expected utility starting at s, first taking action a, and then following policy π

Utility:

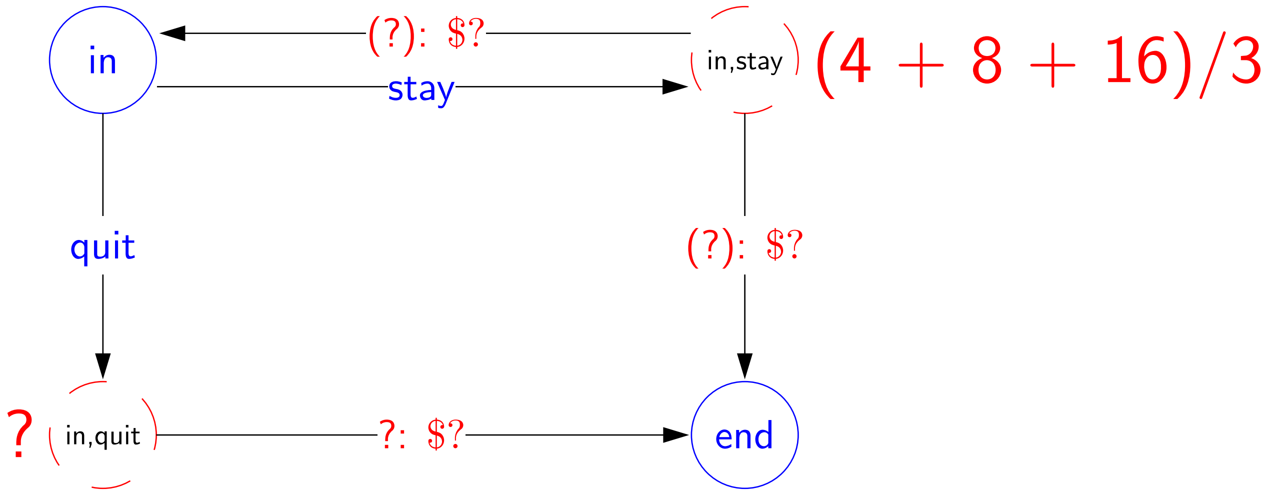
$$u_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots$$

Estimate:

$\hat{Q}_\pi(s, a) =$ average of u_t where $s_{t-1} = s, a_t = a$

(and s, a doesn't occur in $s_0, \cdots, s_{t-2}$)

Model-free Monte Carlo



Data (following policy $\pi(s) =$ stay):

[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]

Note: we are estimating Q_π now, not Q_{opt}

Definition: on-policy versus off-policy

On-policy: estimate the value of data-generating policy

Off-policy: estimate the value of another policy

MDPs: SARSA

Using the reward + Q-value

Current estimate: $\hat{Q}_\pi(s, \text{stay}) = 11$

Data (following policy $\pi(s) = \text{stay}$):

[in; stay, 4, end]	$4 + 0$
[in; stay, 4, in; stay, 4, end]	$4 + 11$
[in; stay, 4, in; stay, 4, in; stay, 4, end]	$4 + 11$
[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]	$4 + 11$

Algorithm: SARSA

On each (s, a, r, s', a'):

$$\hat{Q}_\pi(s, a) \leftarrow (1 - \eta)\hat{Q}_\pi(s, a) + \eta[\underbrace{r}_{\text{data}} + \gamma \underbrace{\hat{Q}_\pi(s', a')}_{\text{estimate}}]$$

Model-free Monte Carlo versus SARSA

Key idea: bootstrapping

SARSA uses estimate $\hat{Q}_\pi(s, a)$ instead of just raw data u.

u	$r + \hat{Q}_\pi(s', a')$
based on one path	based on estimate
unbiased	biased
large variance	small variance
wait until end to update	can update immediately

MDPs: Q-learning

Q-learning

Problem: model-free Monte Carlo and SARSA only estimate Q_π, but want Q_{opt} to act optimally

Output	**MDP**	**reinforcement learning**
Q_π	policy evaluation	model-free Monte Carlo, SARSA
Q_{opt}	value iteration	**Q-learning**

Q-learning

Bellman optimality equation:

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s'|s,a)[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')]$$

Algorithm: Q-learning [Watkins/Dayan, 1992]

On each (s, a, r, s'):

$$\hat{Q}_{\text{opt}}(s,a) \leftarrow (1-\eta)\underbrace{\hat{Q}_{\text{opt}}(s,a)}_{\text{prediction}} + \eta\underbrace{(r + \gamma \hat{V}_{\text{opt}}(s'))}_{\text{target}}$$

Recall: $\hat{V}_{\text{opt}}(s') = \max_{a' \in \text{Actions}(s')} \hat{Q}_{\text{opt}}(s', a')$

Off-Policy versus On-Policy

Definition: on-policy versus off-policy

On-policy: evaluate or improve the data-generating policy

Off-policy: evaluate or learn using data from another policy

	on-policy	**off-policy**
policy evaluation	Monte Carlo SARSA	
policy optimization		Q-learning

Reinforcement Learning Algorithms

Algorithm	**Estimating**	**Based on**
Model-Based Monte Carlo	$\hat{T}, \hat{R}$	$s_0, a_1, r_1, s_1, \ldots$
Model-Free Monte Carlo	$\hat{Q}_\pi$	u
SARSA	$\hat{Q}_\pi$	$r + \hat{Q}_\pi$
Q-Learning	$\hat{Q}_{\text{opt}}$	$r + \hat{Q}_{\text{opt}}$