Reinforcement Learning

Markov Decision Processes

David Abel, Michael Herrmann
Based heavily on slides by Stefano V. Albrecht

21 January 2025

Lecture Outline

- Revisit two questions from last time
- Central formalism: Markov decision processes (MDPs)
- Main quantities, functions: Policies, returns, value functions, Bellman equation.

Revisit Two Questions

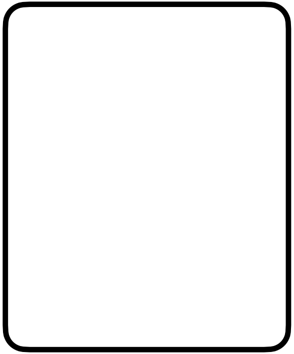
- Q1: Which actions in UCB are explore actions? Which exploit?
- Q2: What is going on with the spike in Fig. 2.3?

Q1: Which actions in UCB are explore actions? Which exploit?

A: Actions can be a mix. Or, either extreme.

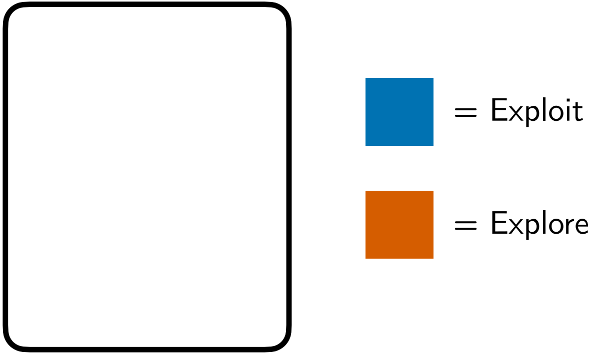
Explore-exploit is about competing *pressures*: get reward *and* learn about the world.

Q1: Which actions in UCB are explore actions? Which exploit?



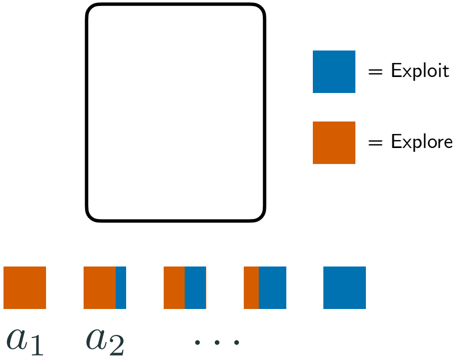
Analogue: Given an empty canvas and a paint brush, paint the canvas 50% orange and 50% blue.

Q1: Which actions in UCB are explore actions? Which exploit?



Analogue: Given an empty canvas and a paint brush, paint the canvas 50% orange and 50% blue.

Q1: Which actions in UCB are explore actions? Which exploit?



Analogue: Given an empty canvas and a paint brush, paint the canvas 50% orange and 50% blue.

Q1: Which actions in UCB are explore actions? Which exploit?

Exploit: Pick best option so far

$$A_t = A_t^* = \arg\max_a Q_t(a)$$

Greedy action selection

Explore: Learn more about other options

$$A_t \sim \mathrm{Unif}(\mathcal{A})$$

Random action selection

Some algorithms *explicitly* divide actions in this way

Q1: Which actions in UCB are explore actions? Which exploit?

Algorithm: UCB

$Q_1(a), N_1(a) = 0, \forall a \in \mathcal{A}$

For each round t in T:

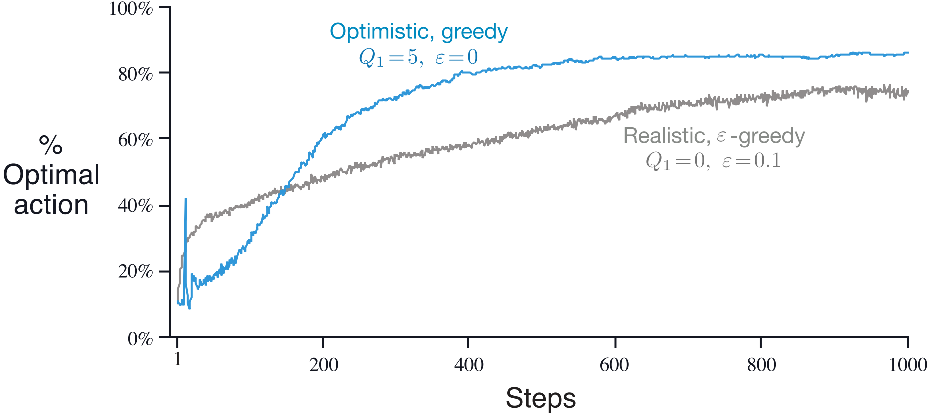
$$A_t = \begin{cases} \mathrm{Unif}(\mathcal{A}) & \max_a N_t(a) = 0 \\ \arg\max_a [Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}] & otherwise \end{cases}$$

Execute A_t, observe R_t

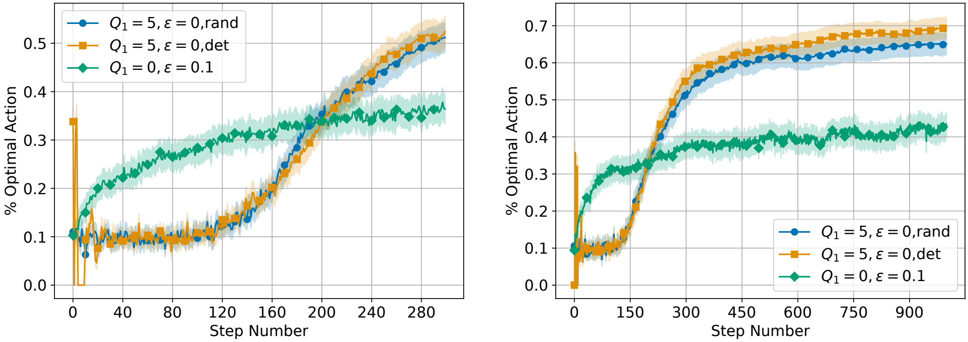
Update $N_t(a)$, $Q_t(a)$

Other algorithms choose actions that *balance* exploration and exploitation.

Q2: What is going on with the spike in Fig. 2.3?



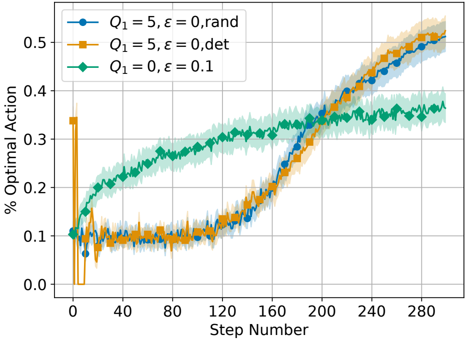
Q2: What is going on with the spike in Fig. 2.3?



(a) First 300 Steps **(b)** Full 1000 Steps

Re-implemented: Blue breaks ties randomly, orange does not.

Q2: What is going on with the spike in Fig. 2.3?



(a) First 300 Steps

```python
if self.break_ties_randomly:
    best_action = random.choice(best_actions)
else:
    best_action = best_actions[0]
```

Blue (no spike)

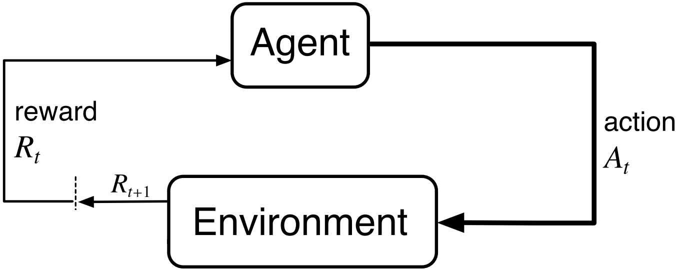
Orange (spike)

(b) The difference in code: blue randomly breaks ties, orange does not.

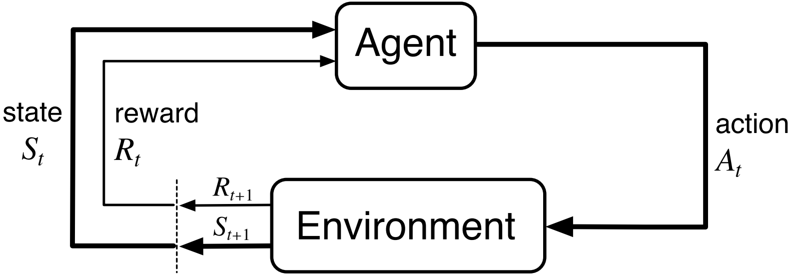
Lecture Outline

- ~~Revisit two questions from last time~~
- Central formalism: Markov decision processes (MDPs)
- Main quantities, functions: Policies, returns, value functions, Bellman equation.

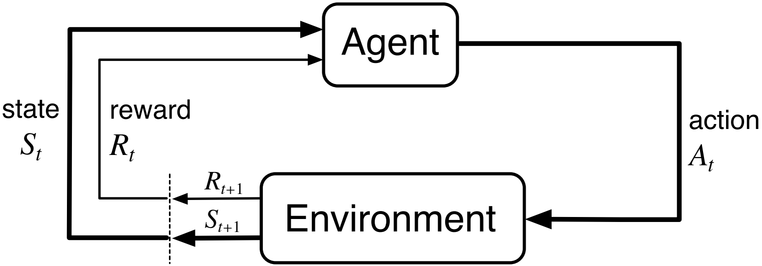
Bandits: The Simplest RL Problem



Bringing State Back: The Agent-Environment Interface



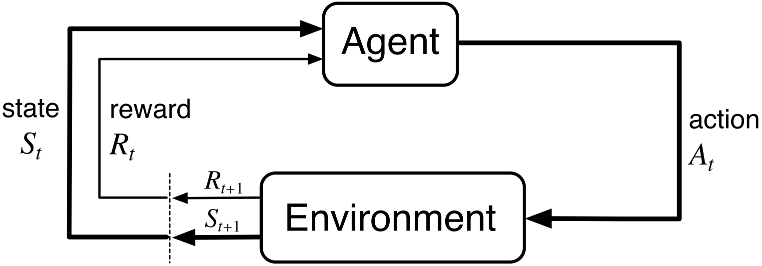
The Agent-Environment Interface

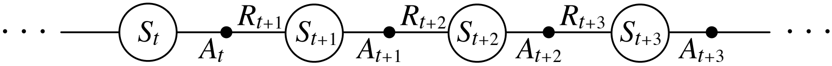


Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

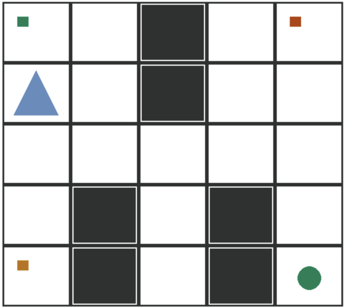
- Agent observes environment state at time t: $S_t \in \mathcal{S}$
- and selects an action at step t: $A_t \in \mathcal{A}$
- Environment sends back reward $R_{t+1} \in \mathcal{R}$ and new state $S_{t+1} \in \mathcal{S}$

The Agent-Environment Interface





Live Demo



Markov Decision Process

Markov decision process (MDP) consists of:

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Reward space $\mathcal{R}$
- Environment dynamics:

MDP is *finite* if $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$ are finite

$$p(s', r|s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$$

$$p(s'|s, a) = \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

$$r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

Markov Property

Markov property:
Future state and reward are independent of past states and actions, *given the current state and action*:

$$\Pr\{S_{t+1}, R_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0\} = \Pr\{S_{t+1}, R_{t+1} \mid S_t, A_t\}$$

- State S_t is *sufficient summary* of interaction history
  $\Rightarrow$ Means optimal decision in S_t does not depend on past decisions
- Designing compact Markov states is "engineering work" in RL

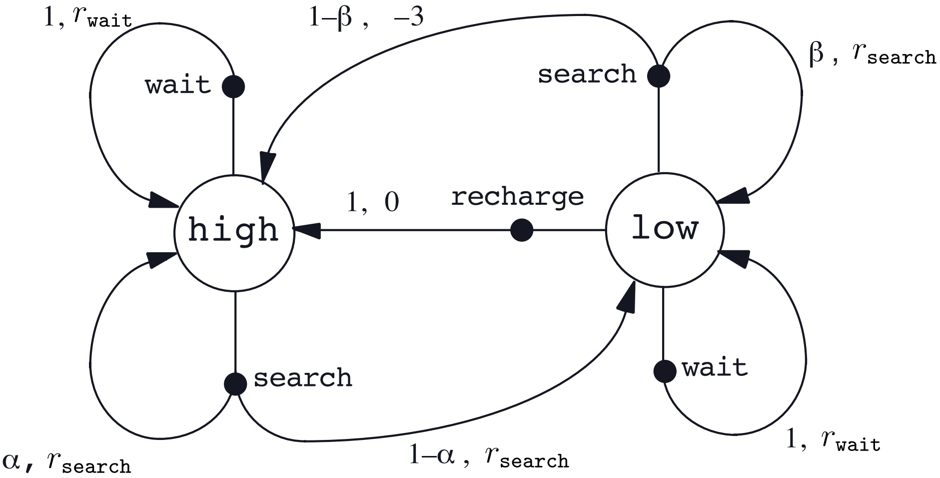
Example: Recycling Robot

- Mobile robot must collect cans in office
- States:
  - `high` battery level
  - `low` battery level
- Actions:
  - `search` for can
  - `wait` for someone to bring can
  - `recharge` battery at charging station
- Rewards: number of cans collected

Example: Recycling Robot

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

Example: Recycling Robot



Policy

See Tutorial 2 & 4

MDP is controlled with a policy:

$\pi(a|s)$ = probability of selecting action a when in state s

$\pi(a\|s)$	search	wait	recharge
high	0.9	0.1	0
low	0.2	0.3	0.5

Special case: *deterministic* policy $\pi(s) = a$

$\pi(s)$
high $\rightarrow$ search
low $\rightarrow$ recharge

Remark: MDP coupled with fixed policy π is a "Markov chain"

Goals and Rewards

Agent's goal is to learn a policy that maximises cumulative reward

Reward hypothesis:

All goals can be described by the maximisation of the expected value of cumulative scalar rewards.

Total Return

Formally, policy should maximise expected return:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$
$$= R_{t+1} + G_{t+1}$$

where T is final time step

Assumes *terminating* episodes:

- e.g. Chess game: terminates when one player wins
- e.g. Furniture building: terminates when furniture completed
- Can enforce termination by setting number of allowed time steps

Discounted Return

For non-terminating (infinite) episodes, can use discount rate $\gamma \in [0, 1)$:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$
$$= R_{t+1} + \gamma G_{t+1}$$

low γ is shortsighted
high γ is farsighted

- e.g. One cookie now, or many later?
- e.g. Financial portfolio management

Discounted Return

For non-terminating (infinite) episodes, can use discount rate $\gamma \in [0, 1)$:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$
$$= R_{t+1} + \gamma G_{t+1}$$

low γ is shortsighted
high γ is farsighted

- Sum is finite for $\gamma < 1$ and bounded rewards $R_t \leq r_{\max}$:

$$\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \leq r_{\max} \sum_{k=0}^{\infty} \gamma^k = r_{\max} \frac{1}{1-\gamma}$$

Discounted Return

For non-terminating (infinite) episodes, can use discount rate $\gamma \in [0, 1)$:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

$$= R_{t+1} + \gamma G_{t+1}$$

low γ is shortsighted

high γ is farsighted

- Sum is finite for $\gamma < 1$ and bounded rewards $R_t \leq r_{\max}$:

$$\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \leq r_{\max} \sum_{k=0}^{\infty} \gamma^k = r_{\max} \frac{1}{1-\gamma}$$

- Definition also works for terminating episodes if terminal states are "absorbing":

  absorbing state always transitions into itself and gives reward 0

Note: This is as far as we got in class on 21 Jan, we will pick up from here next lecture.

State Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

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

Markov: past states/actions don't matter given current state

State Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

Markov: past states/actions don't matter given current state

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

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

State Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

Markov: past states/actions don't matter given current state

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

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|a, s) \left[r + \gamma \mathbb{E}_\pi \left[G_{t+1} | S_{t+1} = s'\right]\right]$$

State Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

Markov: past states/actions don't matter given current state

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|a, s) \left[r + \gamma \mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s'\right]\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

State Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

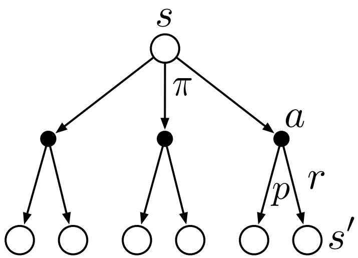
Markov: past states/actions don't matter given current state

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

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|a, s) \left[r + \gamma \mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s'\right]\right]$$

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



One-step look-ahead tree

Action Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

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

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

Action Value Function and the Bellman equation

Because of Markov property, can write state-value function in recursive form with Bellman equation:

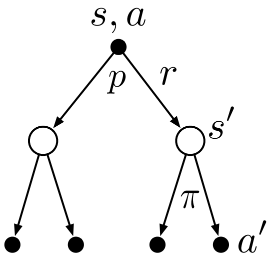
$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

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

Can also define action-value function:

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

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



Recap: Value and Action-Value Functions

value function:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v_\pi(s')$$

Recap: Value and Action-Value Functions

value function:

$$v_\pi(s) = \underbrace{\sum_{a \in \mathcal{A}} \pi(a \mid s) r(s,a)}_{\text{Immediate reward}} + \underbrace{\gamma}_{\text{discounted}} \underbrace{\sum_{s' \in \mathcal{S}} p(s' \mid s,a)}_{\text{expected}} \cdot \underbrace{v_\pi(s')}_{\text{future value}}$$

Recap: Value and Action-Value Functions

value function:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v_\pi(s')$$

action value function:

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v_\pi(s')$$

Recap: Value and Action-Value Functions

value function:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \cdot v_\pi(s')$$

action value function:

$$q_\pi(s, a) = \underbrace{r(s, a)}_{\text{Immediate reward}} + \underbrace{\gamma}_{\text{discounted}} \underbrace{\sum_{s' \in \mathcal{S}} p(s' \mid s, a)}_{\text{expected}} \cdot \underbrace{v_\pi(s')}_{\text{future value}}$$

Optimal Value Functions and Policies

Policy π is optimal if

$$v_\pi(s) = v_*(s) = \max_{\pi'} v_{\pi'}(s)$$

$$q_\pi(s,a) = q_*(s,a) = \max_{\pi'} q_{\pi'}(s,a)$$

Because of the Bellman equation, this means that for any optimal policy π:

$$\forall \hat{\pi} \; \forall s : v_\pi(s) \geq v_{\hat{\pi}}(s)$$

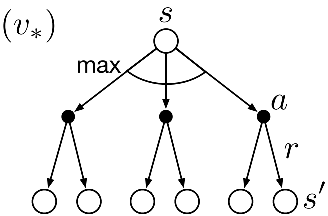
Optimal Value Functions and Policies

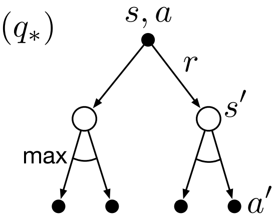
We can write optimal value function without reference to policy:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma\, v_*(s')\right]$$

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right]$$

Bellman optimality equations





Discussion: Relating v_π and q_π

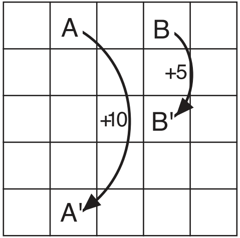
Discussion (2 minutes): Suppose all rewards are non-negative.

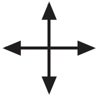
Q: What can be said about the value, $v_\pi(s)$ of a policy π when $\gamma = 0.5$ vs. $\gamma = 0.9$?

Q: When are they equal, if ever?

Gridworld:

- States: cell location in grid
- Actions: move north, south, east, west
- Rewards: -1 if off-grid, +10/+5 if in A/B, 0 otherwise





Actions

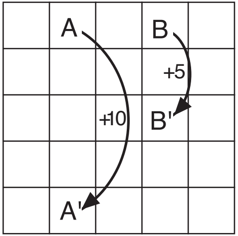
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

State-value function $v_\pi(s)$ for policy $\pi(a|s) = \frac{1}{4}$ for all s, a, with $\gamma = 0.9$

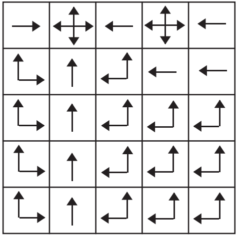
Example: Gridworld

Gridworld:

- States: cell location in grid
- Actions: move north, south, east, west
- Rewards: -1 if off-grid, +10/+5 if in A/B, 0 otherwise



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7



Optimal policy and state-value function

Solving the Bellman Equation

Bellman equation for v_π forms a system of n linear equations with n variables, where n is number of states (for finite MDP):

$$v_\pi(s_1) = \sum_a \pi(a|s_1) \sum_{s',r} p(s', r|s_1, a) \left[r + \gamma v_\pi(s')\right]$$

$$v_\pi(s_2) = \sum_a \pi(a|s_2) \sum_{s',r} p(s', r|s_2, a) \left[r + \gamma v_\pi(s')\right]$$

$$\vdots$$

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

$v_\pi(s)$ are variables

$\pi(a|s)$, $p(s', r|s, a)$, r, and γ are constants

- Value function v_π is unique solution to system
- Solve for v_π with any method to solve linear systems (e.g. Gauss elimination)

Solving the Bellman Optimality Equation

Bellman optimality equation for v_* forms a system of n *non-linear* equations with n variables

- Equations are non-linear due to $\max$ operator
- Optimal value function v_* is unique solution to system
- Solve for v_* with any method to solve non-linear equation systems

Can solve related set of equations for q_π / q_*

Once we have v_* or q_*, we know optimal policy π_* (why?)

Recap: The Main Ideas

- Markov decision process is the canonical way to model RL problems:

$$(\mathcal{S}, \mathcal{A}, r, p, \gamma). \tag{1}$$

- Policy is the agent's strategy for assigning actions to states: $\pi : \mathcal{S} \to \mathcal{A}$ (can be stochastic, too).
- Goal is to find a policy that maximizes *expected cumulative reward*.
- Value: $v_\pi(s)$, Action-value: $q_\pi(s, a)$: capture expected cumulative discounted reward.

RL vs. Planning

- RL problem: efficiently learn a high-value policy by *interacting* with an MDP.
- Planning problem: given an MDP (we know all of its components), compute the optimal policy.

Reading

Required:

- RL book, Chapter 3 (3.1–3.7)

Optional:

- *Dynamic Programming*
  by Richard Bellman (university library has copies)
- *Markov Decision Processes: Discrete Stochastic Dynamic Programming*
  by Martin Puterman (university library has copies)
- Tsitsiklis, J., Van Roy, B. (2002). On Average Versus Discounted Reward Temporal-Difference Learning. Machine Learning, 49, 179–191

[Extra/not examined] Ergodicity and Average Reward

For finite MDP and non-terminating episode, any policy π will produce an ergodic set of states $\hat{\mathcal{S}}$:

- Every state in $\hat{\mathcal{S}}$ visited infinitely often
- Steady-state distribution: $P_\pi(s) = \lim_{t\to\infty} \Pr\{S_t = s \mid A_0, \ldots, A_{t-1} \sim \pi\}$

Performance of π can be measured by average reward:

$$r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid S_0, A_0, \ldots, A_{t-1} \sim \pi]$$

$$= \sum_s P_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\, r$$

Independent of initial state S_0!

[Extra/not examined] Discounting and Average Reward

Maximising discounted return over steady-state dist. is same as maximising average reward!

$$\begin{aligned}
\sum_s P_\pi(s)\, v_\pi(s) &= \sum_s P_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \\
&= r(\pi) + \sum_s P_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[\gamma v_\pi(s')] \\
&= r(\pi) + \gamma \sum_{s'} P_\pi(s')\, v_\pi(s') \\
&= r(\pi) + \gamma \left[r(\pi) + \gamma \sum_{s'} P_\pi(s')\, v_\pi(s')\right] \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 r(\pi) + \gamma^3 r(\pi) + \cdots \\
&= r(\pi)\, \frac{1}{1-\gamma} \qquad \Rightarrow \gamma \textit{ has no effect on maximisation!}
\end{aligned}$$

[Extra/not examined] Discounting and Average Reward

We will focus on discounted return since:

- Most of current RL theory was developed for discounted return
- Discounted and average setting give same limit results for $\gamma \to 1$

  $\Rightarrow$ This is why most often people use $\gamma \in [0.95, 0.99]$
- Discounted return works well for finite and infinite episodes