Reinforcement Learning

Markov Decision Processes

Stefano Albrecht, Pavlos Andreadis

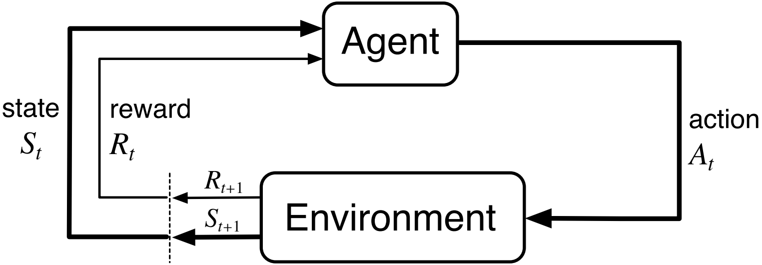
21 January 2020

THE UNIVERSITY *of* EDINBURGH
informatics

Lecture Outline

- Markov decision process
- Policies, goals, rewards, returns
- Value functions and Bellman equation
- Optimal value functions and policies

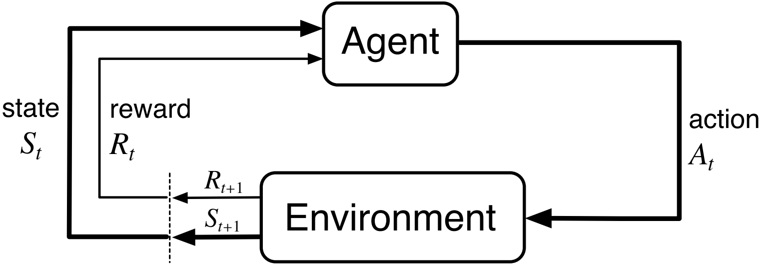
The Agent-Environment Interface

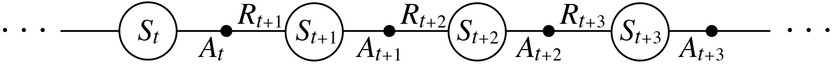


Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

- 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





Markov Decision Process

Markov decision process (MDP) consists of:

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

$\boxed{\text{MDP is } \textit{finite} \text{ if } \mathcal{S}, \mathcal{A}, \mathcal{R} \text{ are finite}}$

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

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

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

Markov Property

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

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

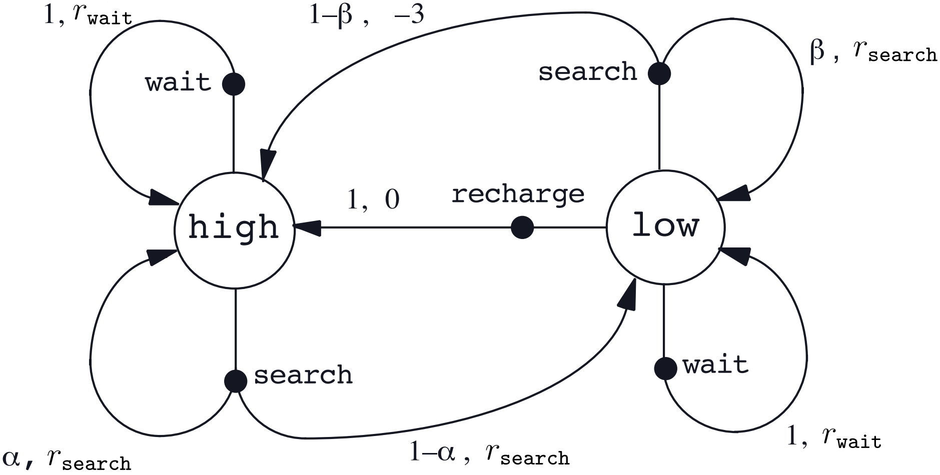
- State S_t is *sufficient summary* of interaction history
  $\Rightarrow$ Means optimal decision in S_t does not depend on past decisions
- Designing compact Markov states is "engineering work" in RL

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

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

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.

Rewards specify *what* the goal is

- Rewards do not specify *how* to achieve goal
- But if done carefully, good reward design may help to learn faster

  $\Rightarrow$ Like state design, reward design is "engineering work" in RL

Total Return

Formally, policy should maximise expected return:

$$
\begin{aligned}
G_t &\doteq R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T \\
&= R_{t+1} + G_{t+1}
\end{aligned}
$$

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 $\gamma \in [0, 1)$:

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

low γ is shortsighted
high γ is farsighted

- e.g. Financial portfolio management
- e.g. Server monitoring and maintenance

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 $R_t \leq r_{\mathbf{max}}$:

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

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

  absorbing state always transitions into itself and gives reward 0

State Value Function

Given policy π, can quantify expected return in any state s with state-value function:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right]$$

State Value Function

Given policy π, can quantify expected return in any state s with state-value function:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s\right]$$

In general, assuming terminating episodes (need more math for non-terminating episodes), let $\mathcal{H}(s)$ be the set of all possible episodes starting in s:

$$\mathcal{H}(s) \doteq \{h = (s^t, a^t, r^{t+1}, s^{t+1}, a^{t+1}, r^{t+2}, s^{t+2}, ..., r^T, s^T) \mid s^t = s\}$$

State Value Function

Given policy π, can quantify expected return in any state s with state-value function:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right]$$

In general, assuming terminating episodes (need more math for non-terminating episodes), let $\mathcal{H}(s)$ be the set of all possible episodes starting in s:

$$\mathcal{H}(s) \doteq \left\{h = (s^t, a^t, r^{t+1}, s^{t+1}, a^{t+1}, r^{t+2}, s^{t+2}, \ldots, r^T, s^T) \mid s^t = s\right\}$$

Each $h \in \mathcal{H}(s)$ has associated probability of occurring and cumulative reward:

$$\Pr(h|\pi) = \prod_{\tau=t}^{T-1} \pi(a^\tau|s^\tau)\, p(s^{\tau+1}, r^{\tau+1}|s^\tau, a^\tau) \quad \text{and} \quad G(h) = \sum_{\tau=t}^{T} \gamma^{\tau-t} r^\tau$$

State Value Function

Given policy π, can quantify expected return in any state s with state-value function:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right]$$

In general, assuming terminating episodes (need more math for non-terminating episodes), let $\mathcal{H}(s)$ be the set of all possible episodes starting in s:

$$\mathcal{H}(s) \doteq \left\{h = (s^t, a^t, r^{t+1}, s^{t+1}, a^{t+1}, r^{t+2}, s^{t+2}, \ldots, r^T, s^T) \mid s^t = s\right\}$$

Each $h \in \mathcal{H}(s)$ has associated probability of occurring and cumulative reward:

$$\Pr(h|\pi) = \prod_{\tau=t}^{T-1} \pi(a^\tau | s^\tau)\, p(s^{\tau+1}, r^{\tau+1} | s^\tau, a^\tau) \quad \text{and} \quad G(h) = \sum_{\tau=t}^{T} \gamma^{\tau-t} r^\tau$$

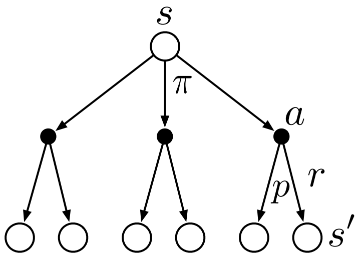
Then compute state value as $v_\pi(s) = \sum_{h \in \mathcal{H}(s)} \Pr(h|\pi)\, G(h)$

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

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



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:

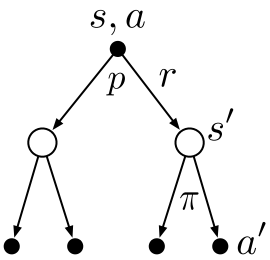
$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

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

Can also define action-value function:

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

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



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:

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

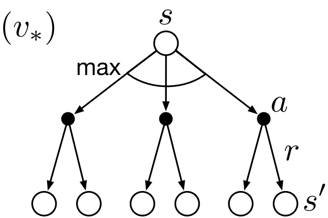
Optimal Value Functions and Policies

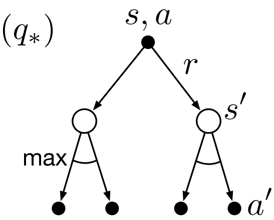
We can write optimal value function without reference to policy:

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

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

Bellman optimality equations

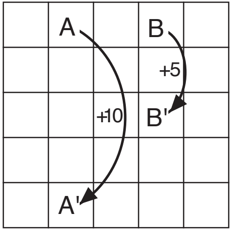


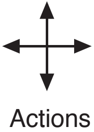


Example: Gridworld

Gridworld:

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





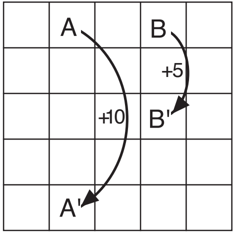
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

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

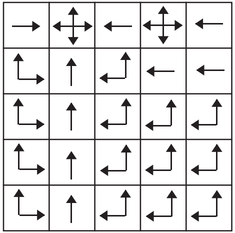
Example: Gridworld

Gridworld:

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

Example: Recycling Robot

Solving for v_* in recycling robot example (states: `h/l`, actions: `s,w,re`):

$$\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{\begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h})+\gamma v_*(\mathtt{h})]+p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l})+\gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h})+\gamma v_*(\mathtt{h})]+p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l})+\gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} \alpha[r_{\mathtt{s}}+\gamma v_*(\mathtt{h})]+(1-\alpha)[r_{\mathtt{s}}+\gamma v_*(\mathtt{l})], \\ 1[r_{\mathtt{w}}+\gamma v_*(\mathtt{h})]+0[r_{\mathtt{w}}+\gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} r_{\mathtt{s}}+\gamma[\alpha v_*(\mathtt{h})+(1-\alpha)v_*(\mathtt{l})], \\ r_{\mathtt{w}}+\gamma v_*(\mathtt{h}) \end{array}\right\}.
\end{aligned}$$

$$v_*(\mathtt{l}) = \max\left\{\begin{array}{l} \beta r_{\mathtt{s}}-3(1-\beta)+\gamma[(1-\beta)v_*(\mathtt{h})+\beta v_*(\mathtt{l})], \\ r_{\mathtt{w}}+\gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array}\right\}$$

Choose numbers for $r_{\mathtt{s}}, r_{\mathtt{w}}, \alpha, \beta, \gamma$ and solve for unique $v_*(\mathtt{h})$ / $v_*(\mathtt{l})$ pair

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!

Discounting and Average Reward

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

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

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

Outlook

- Markov decision process is the fundamental model in RL
- MDPs can be solved exactly if we know all components of the MDP (i.e. $\mathcal{S}, \mathcal{A}, \mathcal{R}, p(s', r|a, s)$)

  $\Rightarrow$ But number of states/actions is problem for scalability
- We will discuss RL techniques which *learn* optimal policy by *interacting* with MDP

  $\Rightarrow$ Methods try to find good *approximate* solutions with reasonable effort

Reading

Required:

- RL book, chapter 3

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